Exploring Trust metrics in a group of Unmanned Aerial Vehicles (UAVs)

Università
di Genova

Elham Mohammadi

DIBRIS - Department of Computer Science, Bioengineering, Robotics and System Engineering

University of Genova

Supervisors:
Prof. Antonio Sgorbissa
Prof. Carmine Recchiuto

In partial fulfillment of the requirements for the degree of

Laurea Magistrale in Robotics Engineering

October 25,2023

Acknowledgements

I genuinely appreciate the invaluable guidance and constant support provided by Prof. Antonio Sgorbissa and Carmine Recchiuto during the course of my thesis. their role as my supervisor, along with their expertise, has been immensely precious to my research. I am profoundly thankful to my parents for their constant support and sacrifices during my academic journey. My fiance's encouragement in challenging times and unwavering belief in my abilities were instrumental in completing this thesis. I'm also grateful to my friends for their steadfast support, wisdom, and shared laughter.

”To all brave women in my country”

Abstract

In an era marked by swift technological progress, new forms of robot interactions have emerged, requiring the integration of different strategies to handle multi-robot collaborative situations. In these scenarios, robots must enhance their cognitive abilities to cooperate efficiently, whether with humans or other autonomous agents. In essence, strengthening the bond and understanding between agents, be it human-robot or robot-robot, hinges upon the exploration of trust concepts and their practical applications.

Given the growing demand for Unmanned Aerial Vehicles (UAVs) in diverse sectors such as agriculture, inspections, security, and surveillance, this thesis aims to evaluate a novel trust framework involving auction-based task assignment to UAVs. It includes overseeing task execution and completion and subsequently establishing agent reliability based on the collected data following the execution and monitoring phase, culminating in an update of trust in the system. In the context of task allocation, our emphasis has been on navigation tasks, specifically the localization of UAVs through Aruco markers and trajectory planning using the A* algorithm. Ultimately, this research offers valuable insights into the importance of trust within an autonomous drone team, particularly in scenarios related to path planning and monitoring.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem statement

The integration of robots into our daily lives has highlighted the need for advanced models of cooperation. When it comes to cooperation in multi-agent systems, the concept of trust among robots becomes important. Earlier studies show that trust is not just about how good the robot is at its job. It also depends on what people expect from the robot in a cooperative context [17] and even how the robot looks [44]. Given the above setting, the need for a robust system to foster communication and cooperation among robots is imperative [13]. Such systems allow them to exchange information, assign tasks, and dynamically adapt to evolving conditions, culminating in enhanced operational outcomes and mission fulfillment. At the core of our thesis lies the investigation and evaluation of a trust-focused framework designed for auction-based task assignments in the realm of Unmanned Aerial Vehicles (UAVs). This customized framework is designed to efficiently oversee a variety of robotic agents, each with unique capabilities. Periodically, these agents receive specific tasks relevant to the context. These tasks are then monitored by other agents, and the system collects data to assess the reliability of these agents, subsequently updating trust in the system. For the purposes of this study, we focused on UAV navigation as the area of interest, addressing two distinct tasks: the self-localization of drones and the localization of other drones, enabling them to observe the behavior of their counterparts. In addition, within this system, we also addressed the task of trajectory planning for Tello drones using the A* algorithm.

Finally, through a network configuration, all UAVs are connected to a central system, enabling them to offer their capabilities when a new task is needed and collectively determine the most suitable agent for the task based on each agent's trust update.

For the purpose of this work, we opted for the DJI Tello drone Edu version due

to its indoor flight stability, good-quality camera, affordable price, and versatile software, which enables it to perform various tasks. However, when we take a closer look, it's clear that there are many tricky technical, technological, and operational problems that need to be solved to get this job done. These challenges are complicated and need careful thought and creative solutions to make sure we can achieve our goal.

In tackling the issue of precisely pinpointing the position of drones within indoor settings and subsequently enhancing their navigational capabilities, we turned to Aruco markers. Their high level of accuracy and ease of use make them perfect for maintaining reliable UAV positioning. Such integration becomes pivotal when the task necessitates swift and precise camera pose detection, as outlined in the work by [14].

Utilizing these Aruco markers, drones are endowed with the capability to autonomously determine not only their own spatial position and orientation but also those of their fellow drones within a predefined area. To facilitate this, markers have been thoughtfully placed throughout the operational space. Moreover, these markers are also attached to every drone's propellers, boosting their ability to navigate effectively.

It's vital to have synchronized control over multiple drones, especially considering the many tasks that demand coordinated actions. In this context, there are multiple frameworks designed for managing drone groups. Within our work, we've chosen to incorporate both TCP and UDP protocols, in tandem with some other networking applications which are further mentioned in detail in 3.5. This integration focuses on simplifying the deployment process while also offering a scalable solution that can adapt to potential future needs.

1.2 Previous work

The use of robots in our daily lives is experiencing rapid growth, with a significant portion of this expansion occurring within manufacturing settings [60]. In particular, as we delve into the era of Industry 4.0 and the Industrial Internet of Things (IIoT), robots are poised to play a pivotal role in revolutionizing modern manufacturing techniques, as highlighted by [24].

However, the integration of robots into our homes is another aspect of this technological surge, as underscored by reports that predict a greater reliance on robots in domestic settings [32]. This extension of robotic applications to everyday life is expected to bring about a diverse range of robots from various manufacturers, all functioning in concert with each other and, at times, in collaboration with humans. This collaborative environment necessitates a critical element: trust. Trust, in the context of robotic cooperation, revolves around the

belief that one robot, when relying on another, has confidence that the latter will perform its tasks with precision and reliability, as elucidated by [6] Therefore, The increasing acceptance of robots in both industrial and domestic environments marks a pivotal shift in human-robot relations. This merging of high-tech solutions and complex collaborations emphasizes the need for trust among robot "teammates" to guarantee smooth and efficient workflows.

One prominent study [32] underscores the importance of trust as a critical metric for task allocation amongst robots via auction-based systems. This research uniquely frames trust by drawing parallels from established models [31] [64] but with modifications catering to a dynamic open environment. In this setting, robots may dynamically enter or exit the network, complete assigned tasks, or verify the tasks executed by fellow robots. Notably, the study delineates between the capabilities of "executing an action" and "verifying the execution of an action", indicating that they do not necessarily overlap. This distinction posits scenarios where a robot may not have the capability to perform a task but can evaluate another robot's proficiency in executing it. Up to now, the developed framework for auction-based task assignment has been mainly tested in simulation and with social robots Pepper and NAO.

When it comes to localizing Unmanned Aerial Vehicles (UAVs), various methodologies have been developed, encompassing techniques like Global Navigation Satellite Systems (GNSS), Visual Odometry (VO), and Simultaneous Localization and Mapping (SLAM). Each of these techniques offers distinct advantages under particular conditions. However, there are instances, such as when the GNSS signals are compromised due to obstructions or interference [55], or when environments present challenges like weak distinguishable features or obstructions that obscure vision [26]. In these complex scenarios, fiducial markers have emerged as a reliable solution. Research, including studies like [39], has indicated that these markers not only offer enhanced accuracy but also ensure robustness, making them particularly suitable for ensuring precise UAV localization under challenging conditions.

Managing teams of drones requires intricate and reliable architectures to ensure smooth operations and effective coordination. As drone technology advances, so does the sophistication of these control systems, with researchers and engineers developing innovative solutions to address the unique challenges of multi-drone management. For instance, the approach detailed in [36] leverages the power and accessibility of cloud platforms. By doing so, it provides a decentralized control mechanism where multiple operators, irrespective of their geographical location, can simultaneously control and monitor multiple UAVs. This cloud-based approach is particularly beneficial as it facilitates scalability, allows real-time data processing, and offers redundancy—qualities essential for large-scale drone operations. Moreover, the cloud's distributed nature means that operators can join

from anywhere with internet access, allowing for flexible and collaborative drone management.On the other hand, the method proposed in [45] harnesses WebSockets, a technology tailored for real-time communication. WebSockets provide full-duplex communication channels, enabling data to flow in both directions simultaneously over a single, persistent connection. Such a communication mechanism is invaluable for drone operations, where real-time feedback and command exchange are crucial. Unlike traditional HTTP connections that open and close for each exchange, WebSockets remain open, allowing for faster and more efficient data transfer, ideal for scenarios demanding instant responsiveness.

1.3 Document structure

Having outlined the main issue and provided an overview of prior research, the subsequent sections of this thesis are organized as follows :

- **Chapter 2:**This section provides a detailed state of the art to the key theoretical concepts crucial for the project. It starts with an extensive look at drone classifications, emphasizing their varied functionalities and potential use cases. The discussion progresses to multi-agent systems, focusing on the cooperative dynamics among multiple drones. Lastly, the importance of navigation and computer vision for drone operations is underscored, emphasizing their roles in movement and visual data interpretation, respectively. This overview sets the groundwork for deeper explorations in the project's subsequent parts.

- **Chapter 3:** provides a deep dive into the software architecture of the system while also shedding light on the hardware components involved. It further clarifies the strategies adopted to address the issues highlighted in 1.1 as the problem statement. This section also encompasses the system's design and its comprehensive execution.

- **Chapter 4:** In this chapter, test results and the outcomes from the physical tests are showcased.

- **Chapter 5:** conclusions and suggestions for future improvements

Chapter 2

State of the Art

2.1 Multi-Agent Systems (MAS)

A Multi-Agent System (MAS) is a coordinated network of agents that interact to achieve specific objectives. These agents can be software entities, hardware components, or a mix of both, and they can vary in their degree of autonomy.

2.1.1 Degree of Autonomy

Although autonomy is often at the heart of many definitions for agents and stands as a key research topic within the multi-agent domain, a universally accepted definition still remains elusive. This paper [7] seeks to categorize the various interpretations of autonomy found in the existing literature by employing the Vowels approach. On the other hand, The article [70] introduces the capability of Dynamic Adaptive Autonomy (DAA), which allows an agent to dynamically modify its autonomy along a defined spectrum (from command-driven to consensus to locally autonomous/master) for each goal it pursues. However, in MAS scenarios, due to the wide range of skills that these agents possess, they often encounter unpredictability, significant variances, and irregular workloads. Such conditions can lead agents to make decisions with unforeseen results. To tackle this issue, this paper [54] presents the Fuzzy Logic-based Adjustable Autonomy (FLAA), aiming to fine-tune the autonomy of multi-agent systems operating in complex environments.

2.1.2 Applications

Multi-agent systems (MAS) have become popular in various areas, acting as a key tool for handling and mimicking complex setups. These systems made up of several cooperating agents, promote shared efforts, flexibility, and independent

problem-solving. In fields like healthcare, robotics, and environment studies, MAS has brought about improvements by encouraging effective teamwork and prompt decisions. Their responsive character, combined with their capacity to manage lots of data at once, makes them essential for tackling the changing challenges we face today.

- **Electric power systems:**The evolution of energy networks from static to dynamic systems necessitates the infusion of intelligence throughout the network. A promising strategy to achieve this is through the use of multi-agent systems technology. In this setup, functional components operate as autonomous agents, communicating and interacting via messaging protocols. This mode of interaction promotes a flexible relationship between components, which can be advantageous for the intricate systems foreseen in the context of the smart grid. These papers [9] [50] take a retrospective look at the pivotal developments and demonstrations of agent-based systems within the power sector.

- **Healthcare:** Multi-agent systems (MAS) comprise groups of self-governing intelligent entities that work collectively to address intricate issues. Given the nature of challenges in the medical field, these systems prove to be particularly relevant and beneficial. this paper [53] posit that MAS is aptly suited for addressing healthcare dilemmas. they delve into specific instances of how this technology has been employed across various health-related challenges, such as the administration of organ transplants, facilitating access to medical data, and caring for palliative patients. Also this literature [10] offers a comprehensive guide detailing the research surrounding the integration of multi-agent systems within the medical healthcare domain.

- **Robotics:**In robotics, MAS plays a crucial role in coordinating teams of robots, enabling them to collaborate efficiently in endeavors such as search and rescue or exploration missions. An illustrative extension of this concept is the use of MAS in multi-unmanned Aerial Vehicle (UAV) systems. These UAVs, operating as a coordinated fleet, can cover large areas, execute intricate tasks [2], and adapt to dynamic environments, all while leveraging the principles of MAS to maintain synchronicity and achieve shared objectives.

2.2 Trust in MAS

Trust holds significant relevance within the domain of autonomous multi-agent systems, where it plays a pivotal role in two distinct subfields: human-robot interaction (HRI) and robot-robot interaction (RRI). These subfields warrant a

more comprehensive examination as they involve intricate dynamics surrounding trust. In HRI, understanding and fostering trust between humans and robots is crucial for effective collaboration and user acceptance. Conversely, in RRI, the establishment of trust among multiple autonomous robots is vital for seamless cooperation and the achievement of collective objectives. Both subfields underscore the multifaceted nature of trust within autonomous multi-agent systems, necessitating nuanced exploration and evaluation.

2.2.1 Human Robot interaction

Numerous studies explore the interaction between humans and robots. For instance, this paper [67] investigates the theoretical basis of trust in the context of human-robot teamwork. The primary objective of this model is to enhance our understanding of the factors that encourage trust between human operators and their robotic partners. The structure of the model is informed by the results of a thorough quantitative meta-analysis. This method categorizes the various dimensions that influence trust in human-robot interaction, including those pertaining to humans, robots, and the surrounding environment.

In a similar vein, the study outlined in [33] suggests that numerous factors play a critical role in assessing and managing trust between humans and robots. These factors encompass attributes related to the robot, such as its performance and physical characteristics, as well as aspects related to humans, including their individual skills and personality traits, along with contextual factors dictated by the specific tasks being performed. To substantiate their hypothesis, the authors conducted a comprehensive analysis of ten scientific articles focused on human-robot interaction (HRI). The findings confirmed the vital importance of trust in collaborations between humans and robots, emphasizing that different factors carry varying degrees of significance. Furthermore, the authors argued that the most influential factors in fostering trust are the inherent characteristics of the robot itself. Conversely, environmental factors exert a moderate influence, while there is limited evidence to suggest that human-related factors have a substantial impact on trust in HRI. Lastly, the results highlighted that establishing a trustful relationship between humans and robots is significantly influenced by several other considerations, including trust calibration and transparency.

2.2.2 Robot Robot interaction

Initially, exploring trust in the context of robots may seem irrelevant, given that robotic agents typically operate within their designated parameters, do not engage in deceitful or malicious actions intentionally (unless compromised, a scenario we won't delve into here), and it's challenging to envision robotic agents

whose judgments about others are influenced by personal emotions or personalities. Consequently, many researchers concentrate on the practical aspects of robot collaboration, emphasizing tasks, cooperative control, and strategic optimization without considering subjective elements like trust. Nonetheless, evaluating trustworthiness doesn't necessarily imply a risk of malevolence. It also becomes crucial when agents must assess their and others' capabilities by comparing prior beliefs and claims to actual performance results. This evaluation, both before and after task execution, becomes vital. As discussed in the introduction, mismatches between a robot's expected and actual capabilities can result from unpredictable environmental conditions, limitations in perception and action, and other factors like incomplete documentation, limited testing, wear and tear, or performance degradation.

Take, for example, the task of allocating assignments to multiple agents. In [42], a comprehensive review of the multi-robot task assignment (MRTA) problem is presented, highlighting various criteria and algorithms tailored to optimize outcomes in various contextual scenarios. When tackling the challenge of distributed multi-robot task allocation, the adoption of auction-based methods [68] emerges as a popular approach. The concept is straightforward, and multiple variants have been proposed [35]. Agents, which may differ in sensory equipment or functional capabilities, bid for tasks, and the auctioneer assigns tasks to the most suitable agent based on predefined metrics.

Regardless of the specific algorithm chosen for task assignment, MRTA invariably relies on shared communication protocols and the assessment of one or more metrics to gauge the suitability of each potential robot for the given task at a given moment. Nonetheless, the judicious selection of evaluation metrics is paramount, given the potential disparity between agents' beliefs and claims and their actual capabilities.

2.3 Unmanned Aerial Vehicles (UAV)

Unmanned Aerial Vehicles (UAVs), commonly known as drones, are aircraft that operate without a human pilot onboard. Instead, they are either remotely piloted, often from the ground or another vehicle, or operate autonomously based on pre-programmed flight plans or more complex dynamic automation systems. These papers [18] [52] provided a detailed survey and comprehensive review of UAVS and their potential applications.UAVs come in various shapes, sizes, and configurations, depending on their intended purpose. Broadly, they are classified into several categories based on different features.

2.4 UAV classifications

Unmanned Aerial Vehicles (UAVs) come in a wide variety of specifications, equipment, sizes, ranges, and shapes. They are available in the market with varying numbers of rotors or propellers. UAVs have been engineered with diverse engines and wing configurations. They are capable of wireless communication over both short and long distances and can be categorized by size, ranging from nano to micro to large. These UAVs hold great potential for providing cellular connectivity due to ongoing advancements in technology. Drones are equipped with a range of features, including First Person View (FPV) goggles, a Global Positioning System (GPS), sensors, stabilizers, and cameras.

2.4.1 fixed-wing UAVs

Fixed-wing UAVs are integral aircraft consisting of wings, a main body, a motor, and a propeller. While they can maintain vertical flight for approximately sixteen hours, their inability to move backward, hover, or rotate can limit their suitability for certain tasks, such as aerial photography. Instead, their design and stability make them ideal for roles like power line inspections and aerial mapping. In line with this, the study presented in [58] aims to provide guidelines for optimizing the aerodynamics and performance of fixed-wing UAVs, especially those operating within the low-speed subsonic range.

Figure 2.1: an example of a fixed-wing UAV

2.4.2 single rotor

Single-rotor UAVs come with their own set of challenges. They are mechanically complex, with intricate components required for rotor control and stabilization [69]. This complexity can lead to higher maintenance and operational costs. Additionally, their design makes them susceptible to vibrations, which can affect the quality of captured data, especially in applications such as aerial photography or mapping.

Due to their mechanical complexity and the need for skilled operators, single-rotor UAVs are often reserved for specialized missions where their unique capabilities are essential. These missions may include search and rescue operations, high-altitude surveys, or tasks requiring heavy lifting capabilities.

Figure 2.2: an example of a single-rotor UAV

2.4.3 multirotor

The most cost-effective and easily constructed UAVs are multirotor UAVs. These UAVs are frequently used in imaging, video surveillance, and transmission line operation and maintenance, for example, this paper [34] delves into the safety inspection protocols and automated detailed examination techniques for transmission towers, utilizing multi-rotor unmanned aerial vehicles (UAVs), they can take the form of quadcopters, hexacopters, or octocopters. Quadcopters have gained immense popularity for their unique attributes, primarily their capacity for vertical takeoff and landing, swift maneuvering capabilities, exceptional agility, and simple yet efficient design. These features make them a preferred choice in various applications due to their cost-effectiveness and compact size. In this regard, their utility extends to a wide range of applications, one of which is discussed in the paper [5] which explores the implementation of deep learning

techniques for fast object detection using quadcopter drones. This technology showcases the adaptability and versatility of quadcopters in the realm of aerial surveillance and data acquisition.

Additionally, another noteworthy paper referenced as [28] introduces a prototype system that harnesses DIY quadcopter drones for product delivery. This innovative approach exemplifies the expanding role of quadcopters in logistics and transportation, where their compact design and maneuverability are harnessed to create efficient and cost-effective delivery solutions.

Furthermore, in yet another work [59], quadcopters are employed for enhancing safety and security measures in the context of building construction sites. This application showcases their potential in surveillance and monitoring, where they can provide valuable insights and enhance situational awareness, ultimately contributing to safer and more efficient construction operations.

In summary, the versatility of quadcopters is evident in their diverse applications, ranging from deep learning-enabled object detection to product delivery systems and safety monitoring in construction sites. Their unique design and capabilities continue to drive innovation in a variety of fields.One example of this kind of drone is the Dji Tello drone which we used for this work.

2.5 Navigation in Robotics

A mobile robot, functioning as a sophisticated intelligent system, must be equipped to continuously monitor its surroundings, accurately interpret its operational environment, strategically map out its path, and make informed decisions based on the data it gathers [56]. The underlying robotic control architectures play a pivotal role in this process. They lay out the blueprint for how these varied capabilities are harmonized and sequenced, ensuring that the robot can seamlessly navigate without human intervention [69]. This integration of sensory perception, environmental interpretation, path planning, and decision-making, steered by the right control architecture, is crucial for the robot's successful and efficient autonomous movement. However, unlike stationary robots, drones, or Unmanned Aerial Vehicles (UAVs), operate in a three-dimensional space with diverse challenges ranging from aerial obstacles to atmospheric changes. Given this complexity, drones require an even more sophisticated level of navigation capability.

To illustrate the advancements in UAV navigation, several works have emerged, addressing specific methodologies and challenges. For instance, this work [73] delves into the realm of swarm drone navigation. Here, an autonomous swarm of drones functions as a multi-agent system. In this setup, the leader agent is imbued with intelligent decision-making capabilities, while the other agents in

the swarm are designed to follow the leader without individualized processing. Such a system showcases how different navigation strategies can be distributed across a group of UAVs to achieve collective objectives.

Conversely, another significant piece of research is documented in [49], offering a comprehensive literature review concerning vision-based methods tailored for UAV navigation. This work underscores the crucial components of visual navigation including visual localization and mapping, obstacle detection and avoidance, and the intricacies of path planning. The ability of a drone to visually interpret its surroundings and make informed decisions is pivotal for safe and efficient navigation, especially in environments where traditional GPS or sensor-based navigation might be challenged. Navigation in UAVs can be broadly divided into three main components:

2.6 Localization

Localization is about determining the robot's position in its environment. This could be relative to the start position or absolute within a map of the environment. However, when it comes to unmanned aerial vehicles (UAVs), their ability to execute autonomous flights is paramount for the success of their designated missions. Achieving this autonomy necessitates a UAV's constant awareness of its precise location. Once the drone discerns its current location, it then formulates navigation directives based on both this immediate position and the predetermined end goal. This information then pilots the drone towards its intended target or endpoint. In outdoor settings, this process of localization becomes relatively straightforward, primarily relying on the synergy between Global Positioning System (GPS) signals and the drone's intrinsic inertial measurement units (IMUs) [30]. However, this methodology encounters significant challenges when transposed to indoor arenas or areas where GPS signals are either weak or entirely non-existent. Addressing this challenge, the research papers [21], [20], and [19] put forth innovative indoor localization strategies, aiming to provide drones with a high degree of positional accuracy even in environments devoid of conventional GPS signals.

2.6.1 common methods for UAV's localization

- **Simultaneous Localization and Mapping (SLAM)** Simultaneous Localization and Mapping (SLAM) [3] is a fascinating and challenging aspect of robotics. It involves a robot moving through an unknown area, trying to figure out its position (localization) while also mapping the space around it. This situation is tricky: a robot needs a map to determine its location, but

to create that map, it must know where it is. To solve this problem, many algorithms and methods have been developed, using information from various sensors like cameras, LiDARs, or sonars. Modern SLAM approaches can handle large areas, detect when they've returned to a previously visited location (loop closures), and skillfully avoid moving obstacles.

Building on this, it's important to mention that while SLAM has made significant progress, traditional GPS/INS navigation systems still face challenges. The numerous advantages of visual navigation—including its vast data, accuracy, minimal disruption, and quick real-time response have naturally drawn the interest of researchers. Exploring this further, the work [72] details the basic concepts and key technologies behind visual SLAM. It also critiques its real-time performance and reliability issues, particularly for UAVs. This paper [11] also highlights an exciting combination, using the aerial view from drones to enhance the limited view of ground robots. By emphasizing a joint mapping and navigation approach based on UAV visual SLAM, the research takes advantage of drones' wide visual range.

- **AR tags**: This article [55] introduces a technique for determining the position of drones in indoor construction settings where GPS is unavailable. It utilizes April tags, which are associated with already established coordinates in the 3D Building Information Model (BIM). By having cameras on the drone and determining the transformation between the tag and the camera, the drone's location within the construction area is identified. This information then guides the drone as it navigates between essential points on the construction site.

  Accurate positioning is vital for the self-guided movement and management of Unmanned Aircraft Systems (UAS). Typically, location data comes from systems like GPS, Galileo, or more modern methods like Visual Odometry, Visual-Inertial Odometry, and SLAM techniques. However, in scenarios demanding precise actions, fiducial markers come into play. These markers provide dependable position data and are employed in numerous applications where reliable pose data is essential for specific items or spots. This study [38] conducts a comparative analysis of four open-source fiducial markers extensively applied in UAS operations: ARTag, AprilTag, ArUco, and STag. These markers' localization proficiency and computational performance form the basis of evaluation.

  In the study [16], a positioning mechanism for UAVs using multiple cameras and ArUco fiducial markers is introduced. Instead of the usual Global Navigation Satellite System (GNSS) or mobile network positioning services, this system leverages multiple optical cameras and ArUco markers integrated

into the UAV structure. The design is versatile, allowing a varying number of cameras to identify an array of drones. A stationary system assessment was conducted, wherein cameras set on the ground viewed the target from varying distances. The gathered data indicates an encouraging precision level for the system.

2.7 Path Planning

Path planning, in general, consists of finding a sequence of actions that transform some initial state into a desired goal state. The states represent agent locations, and transitions between states signify actions the agent can undertake, each associated with a particular cost. [23] Outlines a set of newly formulated heuristic-driven algorithms for real-world route design. As technology evolves, there's a shift towards robots and automated systems operating at swifter rates to cut down on production times [27].To ensure accurate navigation, UAVs need a detailed environmental map or schematic for efficient decision-making. [1] delves into the evolution of UAV route design methodologies over the years.

2.7.1 Some approaches in trajectory planning

- **Dijkstra's Algorithm**: Proposed by Edsger W. Dijkstra in 1956, this algorithm guarantees finding the shortest path in a weighted graph. It incrementally builds a set of nodes that have a minimum distance from the start. As an example [46]proposed a path planning method for smart cars based on the Dijkstra algorithm and dynamic window approach

- **A* Algorithm**: Introduced by Peter Hart, Nils Nilsson, and Bertram Raphael in 1968, the Astar algorithm combines the benefits of Dijkstra's algorithm with heuristics to improve efficiency. It uses a cost function and a heuristic to prioritize nodes that seem more promising.

2.8 Network and control

Controlling multiple drones in a synchronized manner is crucial for numerous applications. There are several architectures for controlling teams of drones, for example, [36] utilizes cloud platforms to allow multiple operators from different locations to control and monitor multiple UAVs simultaneously, or [45] uses WebSockets for full-duplex communication channels over a single, long-lived connection, suitable for real-time data exchange. However, in this work, we integrate TCP and UDP protocols with packet sender and networking applications

for handling the communication between drone agents to emphasize the ease of deployment and ensure scalability for future applications.

2.9 Computer vision

A multidisciplinary field of study that enables computers to interpret and make decisions based on visual data. Essentially, it seeks to teach machines to process, analyze, and understand images or videos in the same way that humans do, but often at a much higher scale and speed.

2.10 applications and libraries

Computer vision algorithms [71] encompass a wide range of mathematical and computational techniques aimed at enabling machines to comprehend and interpret visual data from the world. These algorithms emulate certain aspects of human vision, enabling computers to extract valuable information and glean meaningful insights from images and videos. Image processing algorithms are fundamental and adept at enhancing image quality through noise reduction, sharpening, and contrast adjustments. Feature detection and extraction algorithms [66]identify crucial points in images, like edges and corners, serving as landmarks for subsequent analysis. These algorithms find diverse applications across industries, from healthcare, where they assist in medical image analysis and diagnosis, to automotive, enabling autonomous vehicles to perceive and navigate their surroundings safely. In the field of agriculture, computer vision algorithms aid in crop monitoring and yield prediction [48], while in manufacturing, they enhance quality control and process optimization. Furthermore, retail uses these algorithms for inventory management, shelf monitoring, and cashierless checkout systems. In summary, computer vision algorithms are important tools with wide-reaching applications, revolutionizing how machines interpret and interact with the visual world.

2.10.1 Autonomous Vehicles

Computer vision plays a pivotal role in advancing various aspects of autonomous vehicles. It empowers these vehicles with the capability to detect and recognize objects, such as pedestrians and road signs, ensuring safety. Computer vision also aids in lane detection, traffic sign recognition, and monitoring of traffic conditions, enabling compliance with traffic rules and efficient navigation. Moreover, it contributes to obstacle avoidance, mapping, and localization, crucial for precise and safe autonomous driving [22]. In addition, computer vision enhances driver

monitoring [63] [15], interior safety, and performance in adverse conditions like low light or adverse weather. It extends its benefits to urban mobility and delivery services, enabling autonomous vehicles to navigate complex urban environments and support last-mile deliveries. Furthermore, computer vision facilitates remote supervision and fleet management, making autonomous vehicle operations more efficient and secure, ultimately advancing the field of autonomous transportation across various domains.

2.10.2 AR (Augmented Reality) tags

AR markers or fiducial markers, are physical markers designed to be recognized by AR systems, cameras, or AR applications to augment the real world with digital information or objects. These markers are typically printed on paper or displayed on screens, and they contain patterns or symbols that can be easily detected and tracked by AR software.

AR tags come in various forms, including QR codes, grid patterns, circular patterns, and custom symbols. The choice of marker design depends on the specific requirements of the AR application and the capabilities of the tracking system being used. some of the applications are listed below:

- **Tracking**: This paper [40] introduces an augmented reality (AR) application that utilizes QR Codes for tracking and identification of objects. Instead of traditional AR markers, which are solely used for tracking and don't convey additional information, the proposed system uses QR Codes due to their high information capacity. When the QR Code is scanned, it reveals a 3D virtual representation of the embedded information. The primary application discussed is a product demo system, where a QR Code on a product's packaging displays a 3D virtual object, providing customers with an interactive and direct visualization of the product.

- **Interaction**: AR tags can be used as interactive elements in AR applications [8]. Users can point their device's camera at a marker to trigger specific actions, such as displaying additional information, launching animations, or initiating games.

- **Localization:** In robotics and navigation, AR tags can be placed in the environment to help robots or autonomous vehicles localize themselves accurately [41]. The robot can detect and identify these markers to determine its position within the surroundings, in our case we used ARuco markers to fulfill our purpose.

2.10.3 OpenCV

It is a seminal library in the domain of computer vision [4]. Written in C++ and optimized for real-time applications, it offers interfaces for multiple languages, including Python, Java, and Some core functionalities and applications of OpenCV include:

- **image processing** OpenCV is a powerhouse for image processing, offering a comprehensive suite for different processing needs. Users can efficiently manipulate images by resizing, cropping, or rotating, and can even transform images as required. The library also champions versatility in handling color spaces, supporting seamless conversions between formats like RGB, HSV, and Grayscale. For enhancing image contrast, OpenCV provides tools to analyze and equalize histograms.

  When it comes to refining and enhancing image details, the library offers an array of filters such as the Gaussian process to extract key features of images taken in low light conditions [47]. Morphological operations, encompassing erosion, dilation, opening, and closing, are essential for emphasizing image features, especially in binary representations. OpenCV is also adept at contour detection and analysis in binary images [43], which is pivotal for tasks like shape recognition.

  Other essential capabilities include adaptive thresholding, Canny edge detection, and Sobel operators for image segmentation and edge discernment. Additionally, OpenCV provides robust segmentation techniques like watershed and grabCut to isolate objects and regions in images.

- **Feature Extraction and Matching** OpenCV boasts an extensive set of tools for feature extraction and matching. Among the feature detectors, it offers FAST, SIFT (Scale-Invariant Feature Transform), and SURF (Speeded-Up Robust Features) [51] for example.

  These are specialized algorithms crafted to pinpoint and delineate local features in images, which are crucial for objectives like object recognition. Once key points are identified, OpenCV extends support through descriptor extractors such as SIFT, SURF, and ORB. To ensure these features are properly matched across different images, the library provides algorithms like the Brute-Force matcher in this article for automatic self-checkout system [25] and the FLANN-based matcher [37]for FAST feature matching. These facilitate the recognition of corresponding features in varied images based on the derived descriptors.

- **Video Analysis** OpenCV can read video streams directly from devices,

files, or IP cameras. It supports various file formats, including AVI, MP4, and MKV. You can also write and save videos after processing.

- **Camra Calibrartion** Camera calibration using the OpenCV library is an essential process in computer vision and photography. It involves capturing images of a known pattern, like a chessboard, from various angles. OpenCV's functions automatically detect and locate the pattern's corners in these images, allowing for the identification of corresponding points.

  The library then calculates the camera's intrinsic (focal length, distortion) and extrinsic (position and orientation) parameters, crucial for undistorting and accurately measuring objects in images. Once calibrated, these parameters can be applied to correct distortions in future images captured by the same camera, facilitating precise tasks like object recognition and 3D reconstruction in various applications, from robotics to augmented reality.

- **Aruco Markers** Fiducial markers generally play a crucial role in computer vision, particularly in spatial orientation and location tracking. OpenCV extensively supports these markers. It offers modules for generating and detecting ArUco markers, square binary patterns that facilitate rapid detection and pose estimation. Additionally, OpenCV can identify boards composed of multiple ArUco markers for 3D positioning, calibrate cameras using ArUco grids, and compute the marker's position and orientation relative to the camera, essential for augmented reality (AR) applications.

Chapter 3

System architecture

3.1 system overview

The System Overview section of this thesis report provides a comprehensive introduction to the key components and fundamental structure of the system under study. In this section, we aim to establish a clear understanding of the system's architecture, its main functionalities, and how it operates within its intended environment. This serves as the foundational knowledge required for the subsequent chapters, which delve into various aspects of the system's design, implementation, and evaluation

3.1.1 Trust framework and how it works

For this work, we're exploring and enhancing a novel auction-based task assignment framework developed at DIBRIS. This system is designed for a wide-ranging and open setup, effectively managing different robotic agents. Each agent in the framework is represented by a node, which provides the following services:

- management of auctions either as an auctioneer or a bidder;
- handling of ROS messages through which the agents receive and elaborate data from other agents;
- synchronization with other agents through all the phases of an auction;
- processing of all the data required for the computation of Reliability and Verification Trustworthiness, which are explained further in 3.1.2.
- evaluation of the Perceived Competence , mentioned in 3.1.2 depending on the behaviour-disposition configuration of the agent;

- sending/receiving data, through a TCP/IP socket, to the Adapter to perform the required sensorimotor routines for executing or verifying an action.

The fundamental concept of how this framework operates can be summarized as follows. First, using a portfolio of trust-related metrics, agents dynamically gather data about the other agents' capability to (i) perform actions; and (ii) verify the outcomes of actions performed by other agents. Then, they will iteratively use and update these metrics to evaluate the trustworthiness of other agents, including themselves, during auctions, thus ultimately taking trustworthiness into account when making a new decision for task assignment. To illustrate these concepts, let's consider a simple scenario. Imagine we have a drone named G1 tasked with handling an event called E1, specifically "PerformPathPlanning." This event is initiated by the framework instructions. G1 is equipped with the knowledge and a plan A1, which is planning the path to a certain waypoint, and it is actively involved in the framework. However, G1 recognizes its limitations, as it often encounters difficulties and failures while performing the task.

Then, a noteworthy incident occurs during an auction for A1. A new agent, G2, enters the scene. G2 is a high-end drone with advanced path-planning capabilities and is connected to the local network. G2 expresses a high level of confidence in its ability to perform this particular task. After gathering comprehensive information, including inputs from other participants, G1, acting as the auctioneer, ultimately decides to assign A1 to G2. This decision is made because G2 confidently claims to be exceptionally reliable, and there's no way for any agent, including G1, to dispute this assertion. In stark contrast, everyone is well aware of G1's previous struggles and unreliability in performing this task. Next, G1 gets ready to observe and assess G2's performance, along with other agents who volunteered to do the same. G2 successfully plans its path and reaches the desired destination. However, due to G1's limited vision algorithms, it mistakenly thinks G2 failed. Agents exchange information, and G2 communicates that it completed the task successfully, with other agents in the room agreeing, including the reliable G3 camera, which had a good camera to estimate the pose of agent G2 (or other winners). G1, being the only one to detect a failure, realizes that not only is its ability to perform A1 unreliable but also its ability to confirm if others did it right. As a result, G1 decides to update its beliefs for future instances of event E1.

Once E1 is done, the auction for the second event, E2, begins, involving A2, which is about planning the path to another particular waypoint. G1 wins the auction and starts taking action. This time, G1 accurately assesses the results of the task, and its judgment is supported by most of the verifying agents, including G2 and G3 with their cameras.

It's worth noting that when all agents share their opinions about each other's

success rates in performing and observing actions, decision-making can be a distributed process. Even if there's an auctioneer who assigns the action to the winner, any agent in the framework can predict the auctioneer's choice since they have all the necessary information. This might still be needed for synchronization purposes.

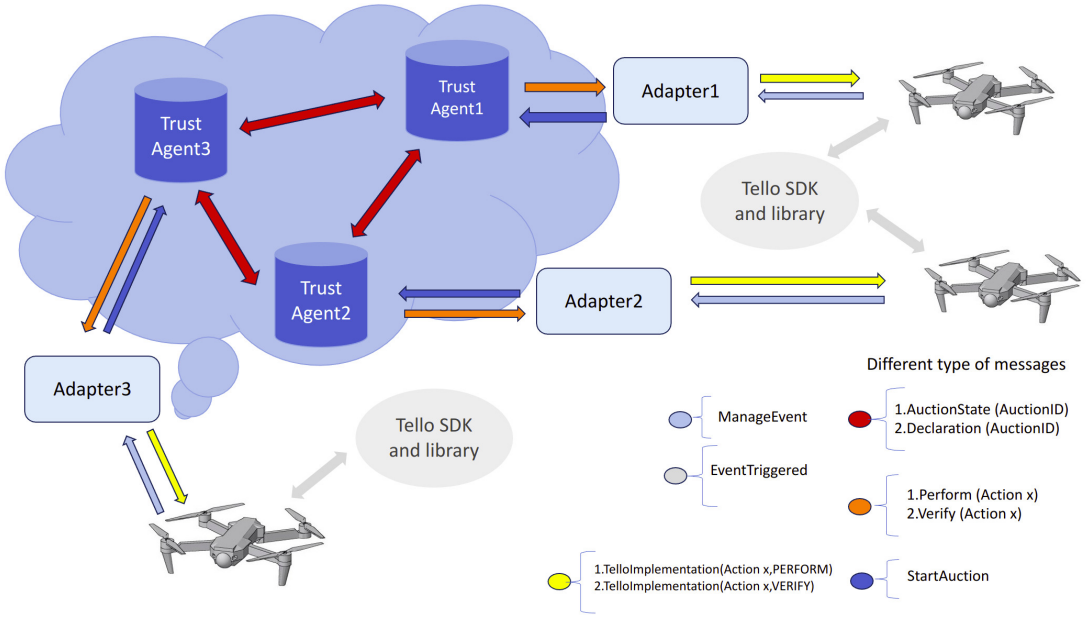

Figure 3.1: Framework architecture

3.1.2 How agent reliability shapes task assignments

The process of assigning tasks to agents in order to carry out plans depends on the use of metrics that help us assess how reliable these agents are. The trustworthiness of agents is a crucial factor in two main aspects: (i) evaluating and ranking bidding agents, which influences the decisions made when assigning tasks, and (ii) considering the opinions of agents regarding the outcomes of these tasks. here we elucidate and explore the overall trustworthiness model from [32], which is also applied in our own research. In the subsequent discussion, we introduce the importance of two context-independent metrics, namely Reliability and Verification Trustworthiness, which are combined to determine an agent's Perceived Competence.

- **Reliability:** Reliability, a constituent of Logical Trust as defined in [12], refers to an assessment made by an agent, denoted as Gi, regarding the

probability of agent Gj's successful execution of a particular action. The concept of reliability is quantified on a scale that ranges between 0 and 1, where 0 stands for failure and 1 for success.

- **Verification trustworthiness:**When an agent carries out a specific action, there is a possibility that other agents may step forward to confirm the results. Consequently, it becomes essential not only to assess the reliability of an agent when it performs a given task but also to gauge how dependable that agent is in confirming the accurate execution of that task. To address this concern we explain the metric "Verification Trustworthiness." This metric quantifies the level of trustworthiness of a confirming agent based on the consensus it garners regarding its judgment abilities. More precisely, it assesses the extent to which the verification provided by agent Gj regarding the success of action Ak is regarded as reliable by agent Gi. This evaluation involves counting the number of opinions that are concordant or discordant from Gj's judgment. The Verification Trustworthiness metric is represented on a scale between -1 and 1. When all agents are in agreement with Gj, it attains a higher value, but as the number of agents expressing discordant views increases, its value diminishes. Ultimately, when all agents disagree with Gj, the metric reaches its lowest point, -1.

- **Perceived competence:** Whenever an auctioneer, represented by Gi, faces the task of determining which bidding agent, denoted as Gj, should be assigned to perform a particular action Ak, Gi engages in a calculation to ascertain the Perceived Competence of Gj. This Perceived Competence is derived through a function that takes into account the Reliability of Gj, which is estimated based on input from all agents participating in the auctions (G1, G2, ..., Gn), as well as their individual Verification Trustworthiness. These relevant values are stored in vectors, and subsequently, the Perceived Competence [57] is calculated utilizing this information.

3.1.3 Framework integration into ROS

The Trust Framework is integrated into ROS [65], which establishes the necessary common language and communication protocol for agents to interact within an open and distributed environment. This framework has been designed with modularity in mind, allowing it to function in both simulated environments and with actual robotic platforms.

A diagram depicting the proposed architecture can be seen in Figure 3.1, It's worth noting that I obtained assistance from the original paper from the University of Genova [32] on this framework when gathering the figures. The

TrustAgent node assumes responsibility for all tasks related to the framework, including communication with other framework members, managing auctions, and evaluating trust. The Adapter, on the other hand, handles the interface with platform-specific sensorimotor routines related to perception and action. This Adapter can be customized to suit various robotic platforms, ensuring both modularity and compatibility.

Diving deep into the intricacies of the framework folder, there lies a multitude of packages designed to cater to various functionalities. While all are significant in their right, two packages stand out due to their distinct roles and capacities, The first one is the situation simulator elaborated as below:

- **Functionality**: This package acts as a virtual environment that mimics real-world scenarios.
- **Event Management**: Users can customize, tune, or modify different events, allowing for diverse simulation scenarios. This adaptability ensures the system is prepared for a variety of situations.
- **Action Control**: Actions within the simulation can be tailored, meaning one can adjust how entities behave and interact within this virtual space.
- **Behavior Triggering**: This module allows for specific system behaviors to be initiated based on the events and actions that transpire within the simulation. It's a feedback mechanism that ensures the system reacts appropriately to various stimuli.

the second package is Trusting Agents:

- **Functionality**: The essence of this package is to foster trust within the system by defining and managing various trust-related parameters.
- **Trust Metrics Definition**: This directory provides the fundamental definitions and formulas that quantify trust within the system. By standardizing these metrics, the system can consistently evaluate and establish trust levels.
- **Auction Data Management**: It maintains a repository of all auction-related information, ensuring that task assignment processes are transparent and can be audited if necessary.
- **Trust Agent Details**: Every agent within the system comes with specific attributes and capabilities. This section outlines these details, ensuring the right agents are assigned the right tasks.

- **launch file**: We've fine-tuned certain parameters in the launch file, which specify the count of events and their corresponding actions, as well as the IP and port settings for each adapter facilitating communication from ROS to external systems. Moreover, we refined settings concerning agent reliability, verification trustworthiness, and their perceived competence toward themselves and other agents.

- **Adapter Information**: Adapters act as intermediaries, bridging gaps between different system components. This directory offers detailed data on these adapters, ensuring seamless integration and communication between various system elements

3.2 DJI Tello EDU drone

DJI Tello is produced by Shenzhen Ryze Technology. This quadrotor incorporates DJI flight control technology and Intel processors (a high-performance vision processing unit based on the Intel Movidius MA2x chipset). It is a nano-size quadrotor (9.8 × 9.2 × 4.1cm). The weight is 80 g including the battery (Li-Po, 1100 mAh, 3.8 V) and 3 " propellers. It provides a maximum flight time of up to 13 min (in no wind conditions with a constant speed of 15 km/h). Using the WiFi signal, the maximum flight distance is 100 meters, and the maximum altitude is 30 meters. The versatile and accessible nature of the DJI Tello drone has made it a popular choice in various research endeavors. One noteworthy application involved the utilization of this drone in a study aimed at identifying human injuries from aerial images obtained during low-altitude flights [29]. This innovative approach demonstrated the potential for drones to play a role in emergency response and remote monitoring, particularly in situations where conventional access might be limited.

In another research endeavor, detailed in a separate paper [62], a sophisticated set of machine learning-driven tools was developed to enhance the capabilities of the OpenMV microcontroller in conjunction with the DJI Tello drone. These tools facilitated precise control of the drone's navigation and enabled it to accomplish specific mission goals. One such successful mission was the onboard detection of individuals wearing protective masks in crowded environments, a task that gained particular relevance during the COVID-19 pandemic.

The DJI Tello's adaptability and compatibility with advanced technologies make it a valuable platform for exploring a wide range of applications in research and development, pushing the boundaries of what can be achieved with consumer-grade drones.

3.2.1 camera and sensors

Tello EDU comes with a 5MP monocular camera that can shoot 720p HD video and has a good camera field of view (FOV) of 82.6°, which is particularly useful for FPV (First Person View) Regarding sensors, it features a Vision Positioning System (VPS) that combines a forward-facing camera and an infrared sensor to ensure stable hovering, even indoors.

Parameter	Value
accelerometer and gyroscope	no details provided
barometer	no details provided
GPS	unavailable
Wifi	2.4 GHz
front-facing camera	electronic image stabilization
FOV	82.6°
video resolution	1280 x 720 pixels (HD,30fps)
image resolution	2592 x 1936 pixels (5Mpix)

Table 3.1: DJI Tello Ryze sensors

3.2.2 hardware components

The computing hardware structure of the system can be broadly divided into two main components. Firstly, the drone is equipped with an onboard flight controller responsible for fundamental functions like stabilizing flight, regulating attitude, controlling translational velocity, and executing basic trajectories. This part of the system operates on closed hardware and communicates using a dedicated command protocol.

Secondly, a higher degree of autonomy can be attained through an external ground-based computer. This external computer leverages telemetry data and video feed to control the drone, utilizing the communication protocol mentioned earlier. This split and distributed computing approach offers the advantage of circumventing size and weight constraints associated with the drone's hardware. Additionally, it allows for the use of a powerful external PC workstation to handle more advanced navigation tasks due to its substantial computing capabilities.

3.3 Drone Localization

As previously discussed in 1.1, the evaluation of the trust framework's performance on a group of Unmanned Aerial Vehicles (UAVs) necessitates the allocation of specific tasks to these drones. To facilitate comprehension within this

section, let's familiarize ourselves with some key terms that will be central to this discussion. As indicated in section 3.1.1, during each iteration of an event, denoted as "Event I," every individual drone is categorized as either a "performer" or an "observer" drone, terms we will continue to employ for clarity.

Within this section, our focus shifts to elucidate the task associated with observer drone verification, which can be divided into two distinct stages. The first stage encompasses drone self-localization using fixed Aruco markers in the environment [1], while the second revolves around localizing the collaborative agent with the help of affixed markers to the body of the drone, commonly referred to as the "performer." To ensure a thorough understanding, we will delve deeper into the intricacies of these processes, accompanied by relevant mathematical equations that underpin their implementation. Through this comprehensive exploration, we aim to shed light on the fundamental mechanisms behind observer drone verification and its significance within the broader context of the trust framework for UAVs.

3.3.1 Spatial Frames in Experimental Setup

As illustrated in figure 3.2, our experiment involved the utilization of four distinct spatial frames: the global frame, the Aruco frame, the camera frame, and the robot frame. These frames can be categorized into three computational layers, which will be explored in more detail in transformation matrices part **??**.In the initial layer, we have successfully ascertained the position and orientation of the Aruco marker frame within the global reference frame. Progressing to the second layer, our goal is to determine the transformation matrix necessary for establishing the position and orientation of the camera frame in relation to the Aruco marker frame. Finally, in the third layer, our primary focus is dedicated to the computation of the rotation matrix that defines the relationship between the camera reference frame and the robot's reference frame, which is centrally located at the robot's center of mass.

3.3.2 drone self-localization

In this section, we embark on a detailed exposition of the algorithm meticulously crafted for the purpose of enabling the observer drone to autonomously determine its precise location within the given environment. This is achieved through the utilization of known Aruco markers, which provide essential data regarding both their positions and orientations. While this process encompasses several integral

[1] refer to my GitHub repository for the source code https://github.com/elh4m/Tello-Aruco-pose-estimation-and-localization

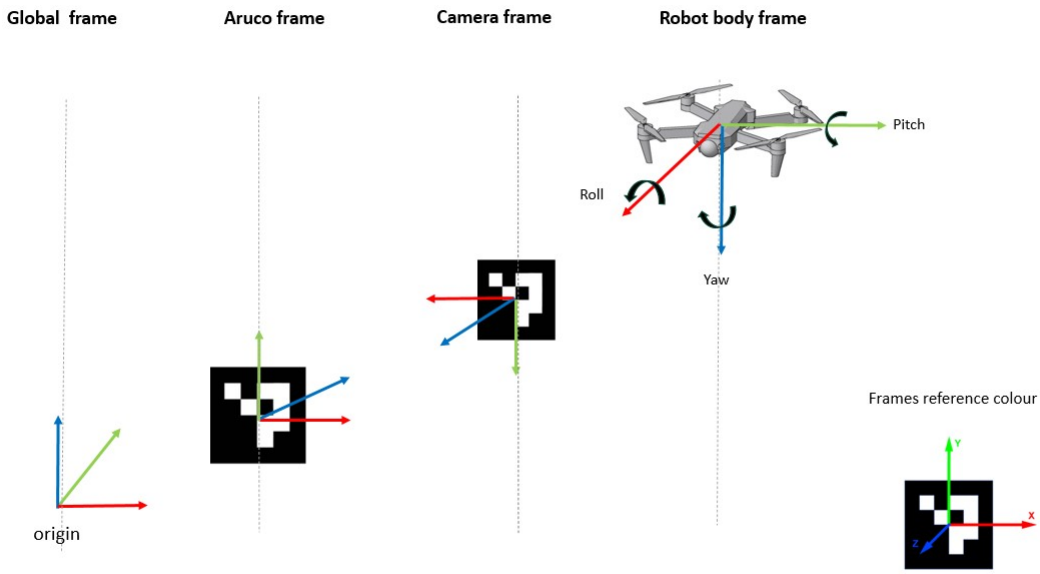

Figure 3.2: all reference frames used during experiment

functions, we will systematically guide you through each step, elucidating their respective roles.

To initiate the algorithm, we strategically deploy an ample number of Aruco markers across the environment, ensuring comprehensive coverage of the drone's camera field of view from all possible angles. Simultaneously, we meticulously define the pose, encapsulating both position and orientation, of these markers within the global reference frame.

Next, we delve into the procedure of accessing the drone's camera, which enables us to detect the Aruco markers and subsequently estimate the camera's pose in relation to the Aruco frame. As we progress through this complex process, we diligently acquire all the necessary transformation matrices at each computational layer.

Finally, we culminate our algorithm by calculating the ultimate output – the position and orientation of the observer drone within the global reference frame. This algorithm is a pivotal component of our study, and its intricacies will be unveiled to provide a profound understanding of the methodology behind observer drone localization within the given environment. This relationship is established through a transformation matrix, denoted as T-cam-Origin. To calculate this

matrix, two sub-matrices are involved:

- **T-aruco-origin**: This sub-matrix represents the transformation from the origin's frame to a fixed Aruco frame. In simpler terms, it defines how the marker is situated within a known reference frame. This transformation matrix is predetermined based on the characteristics of the Aruco marker and its placement.

  To compute this transformation matrix, we employed four distinct rotation matrices. Given the rectangular nature of our environment, the Aruco frame can undergo four distinct rotational variations, in relation to the origin frame of reference, they are provided in these equations 3.1, 3.2, 3.3, 3.4. Furthermore, we determined the translation vector for each individual marker using its unique identifier. Subsequently, all this information was stored in a dictionary, associating marker IDs with their respective positions and orientations.

  Whenever a marker ID is detected, a dedicated function is triggered to examine the dictionary and retrieve the marker's information. This retrieved information is then used for further computations, specifically in the context of calculating the transformation matrix that bridges the Aruco and origin frame. In an environment with a rectangular shape, where the origin is located at the top left corner, the rotations around this origin can be described as follows:

  R1 is the first rotation, occurring as you move clockwise from the origin to the right edge of the rectangular area.

$$\mathcal{R}_1 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.1}$$

  Following R1, the next rotation as you continue clockwise around the rectangular area.

$$\mathcal{R}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.2}$$

  and so on for R3

$$\mathcal{R}_3 = \begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.3}$$

and R4

$$\mathcal{R}_4 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.4}$$

- **T-cam-aruco**: This specific sub-matrix characterizes the transformation from the camera's vantage point to the marker's frame. It effectively encapsulates key details concerning the camera's precise position and orientation in relation to the marker. To derive this sub-matrix, our first step involves the calculation of the transformation matrix between the Aruco marker and the camera, which we refer to as T-aruco-cam. This is achieved through the processes of camera calibration and pose estimation, making use of the results generated by the 'estimatePoseSingleMarkers' function within OpenCV. This function offers us invaluable insights in the form of two essential components: a rotational vector and a translation vector.To proceed with our calculations, we employed the 'Rodrigues' function, which allowed us to derive a rotation matrix from the rotational vector. Once this rotation matrix was obtained, we were fully equipped to compute the transformation matrix.

  Subsequently, to establish the transformation from the marker's frame to the camera's image frame, we compute the inverse of this matrix, yielding T-cam-aruco. This matrix, T-cam-aruco, is pivotal in depicting the transformation from the marker's perspective to the camera's image frame, serving as a crucial element in our overall localization process.

- **T-robot-cam**: Upon marker detection by the camera, our focus shifts to estimating the pose of the robot body frame in relation to the camera. To accomplish this, we introduce an additional transformation matrix known as the T-robot-cam, which plays a pivotal role in achieving this specific objective. This transformation matrix serves as an intermediary in translating information between the image frame and the robot's body frame, enhancing the precision and accuracy of the localization procedure.

- **T-cam-origin**: The computation of T-cam-origin, a critical transformation matrix for extracting the pose of the camera in the global frame and further for calculating the co-agent location, is accomplished by the multiplication of these two sub-matrices (T-aruco-origin and T-cam-aruco). For a visual representation, please refer to Figure 3.3 and the corresponding equation for T-cam-origin.

- **T-robot-origin**: The computation of this transformation matrix, is accomplished by the multiplication of these three sub-matrices. For a visual

representation, please refer to Figure 3.3 and the equation for T-robot-origin, which illustrates the comprehensive transformation process from the robot frame to the origin frame. Once we have this transformation matrix at our disposal, we can extract the drone's orientation within the global frame, which is achieved through the utilization of external Euler angles using function rotationMatrixToEulerAngles(R) from scipy python library.

here you can find the formula for localizing the observer drone :

$$T_{\text{robot\_origin}} = T_{\text{aruco\_origin}} \cdot (T_{\text{cam\_aruco}} \cdot T_{\text{robot\_cam}})$$

$$T_{\text{cam\_origin}} = T_{\text{aruco\_origin}} \cdot T_{\text{cam\_aruco}}$$

$$^{o}T_{r} = {}^{o}T_{a}\,({}^{a}T_{c}\,{}^{c}T_{r})$$
$$^{o}T_{c} = {}^{o}T_{a}\,{}^{a}T_{c}$$

3.3.3 Coexisting agent localization

In this section, we will explore various factors, techniques, and technologies employed to determine the performer drone's location within the given environment. This includes considerations such as sensor data, reference frames, and mathematical transformations. We'll also delve into the algorithms and equations that underpin this calculation, ensuring that you gain a deep insight into the methodology behind this essential aspect of our project.

- **Field of View Checking**: In the initial phase, the observer drone conducts a thorough assessment to ascertain whether the performer drone, which has previously landed on a designated tile, falls within its field of view (FOV). To accomplish this, we've developed a dedicated algorithm specifically designed to calculate the FOV. This algorithm takes into account several critical factors, including the current camera angle, the tile's index, and the overall dimensions of the environment.

  The FOV assessment is a pivotal step in ensuring the observer drone's awareness of the performer drone's presence. It's essential for facilitating efficient communication and coordination between these aerial agents during the execution of their assigned tasks. This algorithmic approach not only enhances the observer drone's situational awareness but also contributes to the overall effectiveness and precision of the collaborative operation.

- **Marker Detection**: If the performer drone is within its FOV, the observer drone proceeds to detect the markers attached to the propeller guard of the performer drone. These markers serve as reference points for determining the performer's location.

- **Location Retrieval**: The observer drone employs the detected markers to retrieve the performer's precise location. Once we've determined the T-cam-origin for the observer drone, the next step involves computing the T-aruco-origin for the agent drone (the performer drone with attached markers to the body frame) to accurately pinpoint its location. A visual representation of these transformations can be found in Figure 3.3.

$$T_{\text{aruco\_origin}} = T_{\text{cam\_origin}} \cdot T_{\text{aruco\_cam}}$$

$$^{o}T_{a} = {}^{o}T_{c}\,{}^{c}T_{a}$$

- **Location Comparison**: Finally, the observer drone compares the retrieved location of the performer with the location that the performer had previously broadcasted. For more comprehensive details about the communication and networking aspects, I recommend referring to Section 3.6.1, as it provides a deeper understanding of how information exchange plays a crucial role in this drone coordination process. If the two locations match, the observer drone returns a "True" result; otherwise, it returns a "False."

3.4 Drone navigation

In this section, we delve into the intricacies of our Tello drone's A* path planning algorithm [1], which plays a pivotal role in determining the most efficient route through obstacles to reach a designated destination and subsequently guides the drone back to its initial position with the utilization of the return-to-home function. This algorithm is meticulously implemented in Python, ensuring seamless integration with the Tello SDK.

The A* path planning algorithm, as employed in our system, initiates by constructing a graph representation of the environment using the "map-graph-generator" function. This graph serves as a structured framework that encapsulates critical information about the spatial layout and obstacles within the environment.

[1] refer to my GitHub repository for the source code https://github.com/elh4m/Tello-drone-Astar-path-planning-python

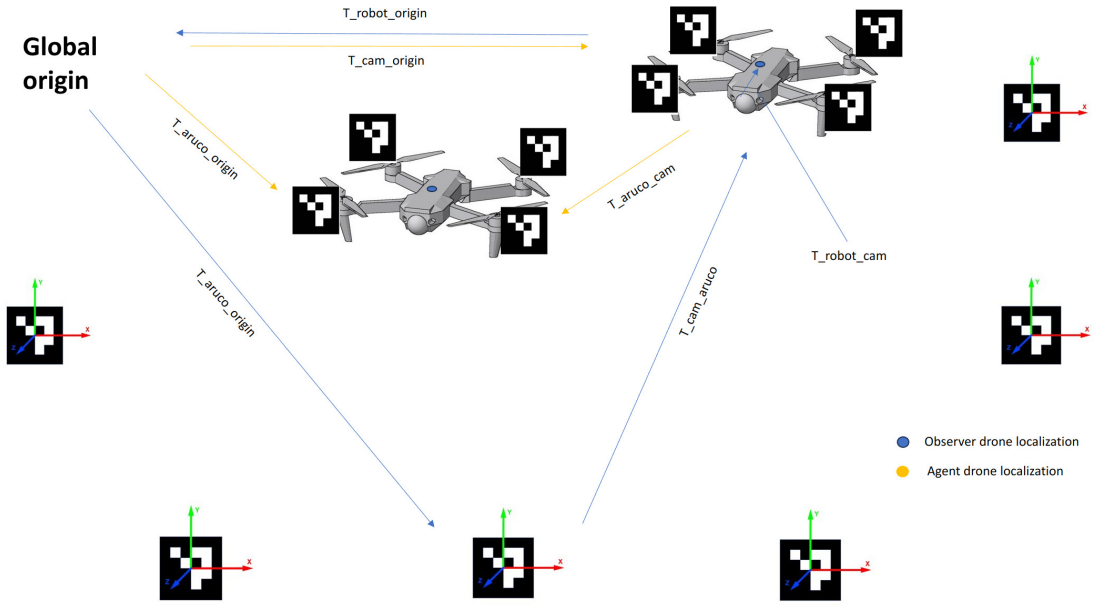

Figure 3.3: Drones transformations matrices

Subsequently, we employ the A* algorithm itself to calculate the shortest path among obstacles, considering factors such as proximity, terrain, and any obstructions in the drone's path. This step is fundamental in determining the optimal route that the drone should follow to reach its intended destination.

Based on the path derived from the A* algorithm, we issue a series of flight commands to the Tello drone. These commands include instructions for takeoff, rotations in a counter-clockwise direction, and precise movements in specified directions. This orchestration ensures that the drone follows the calculated path accurately and efficiently.

To provide a comprehensive understanding of this process, we offer a schematic representation that visually illustrates the functioning of this algorithm. By the end of this section, readers will have a thorough insight into how our Tello drone successfully navigates through its environment, effectively avoiding obstacles and reaching its intended destination while ensuring a safe return to its starting point.

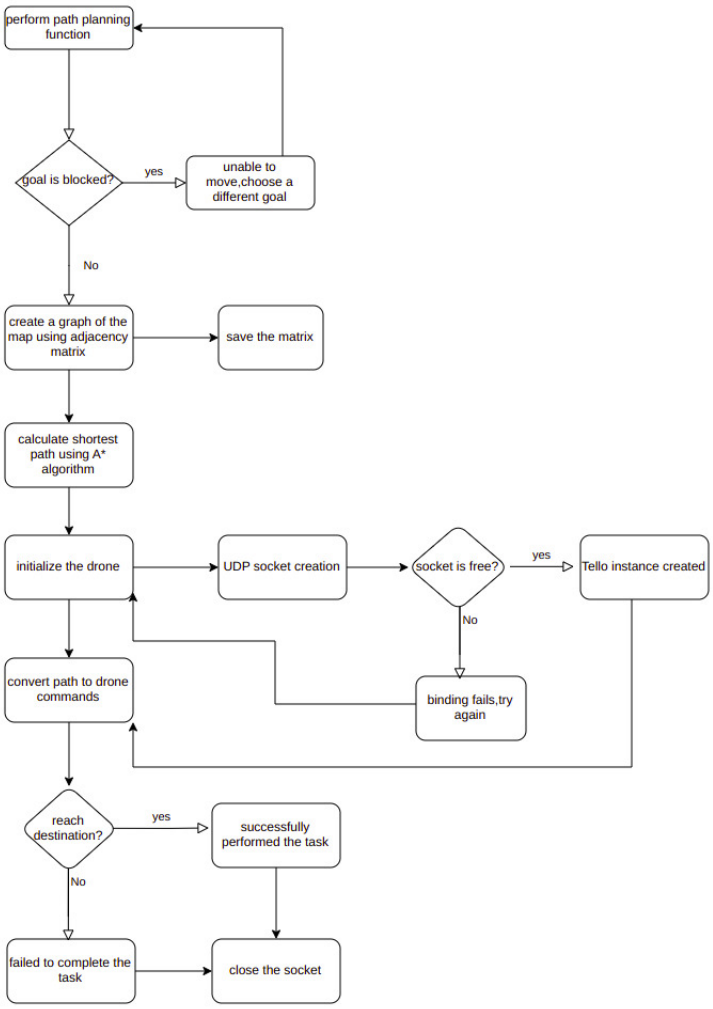

Figure 3.4: Visual exploration of Tello drone A* trajectory planning algorithm

3.5 Software utilization

In this section, we are presenting all system components employed in this project, beginning with the top-level system architecture and extending to the specific applications, along with their respective installed versions.

3.5.1 ROS

ROS (Robot Operating System) doesn't function like a traditional operating system that manages processes and scheduling [61]. Instead, it serves as a structured communication layer above the underlying operating systems of a diverse com-

puting cluster. The core concepts in the implementation of ROS include nodes, messages, topics, and services.

Nodes are like computational processes, fostering modularity within a system, and they communicate through messages, which are structured data entities. These messages can contain various data types and can even be nested.

Nodes share data by publishing messages with no specific topics, and other nodes interested in that data can subscribe to those topics. ROS also supports services, enabling nodes to request specific actions from others in a client-server manner, enhancing functionality.

In our specific scenario, we opted for the ROS Noetic version, which offers enhanced robustness. This version was installed directly on a native Ubuntu 20.04 system. This choice was made to ensure the durability and stability of our ROS environment.

3.5.2 websockets

WebSockets [1] is a communication protocol enabling bidirectional, real-time interaction over the web. Traditionally, the web-operated request-response paradigms where the client requests and the server responds. WebSockets revolutionize this by allowing both client and server to send data at any time, making live updates and interactive applications, like online gaming and chat systems, viable without constantly polling the server.

Beginning with an HTTP handshake, the protocol establishes a persistent connection, replacing repetitive open-close cycles, and reducing overhead and latency. The fundamental communication framework that connects a ROS-based server to an adapter, which is implemented as a Python script utilizing the TCP communication protocol, heavily relies on the Python socket module. This module plays a crucial role in ensuring real-time data exchange with minimal latency, facilitating synchronized operations.

3.5.3 packet sender application

The primary method to connect the Tello Edu drone to a PC is by broadcasting a Wi-Fi signal. The computer then establishes a connection via SSH. Consequently, at any given moment, only one drone can be linked to a PC (or put another way, one PC can only connect to one drone). Additionally, when a drone is connected, the PC loses its internet connection. For our specific setup, we utilized the Packet

[1] find the full documentation for python socket library https://docs.python.org/3/library/socket.html/

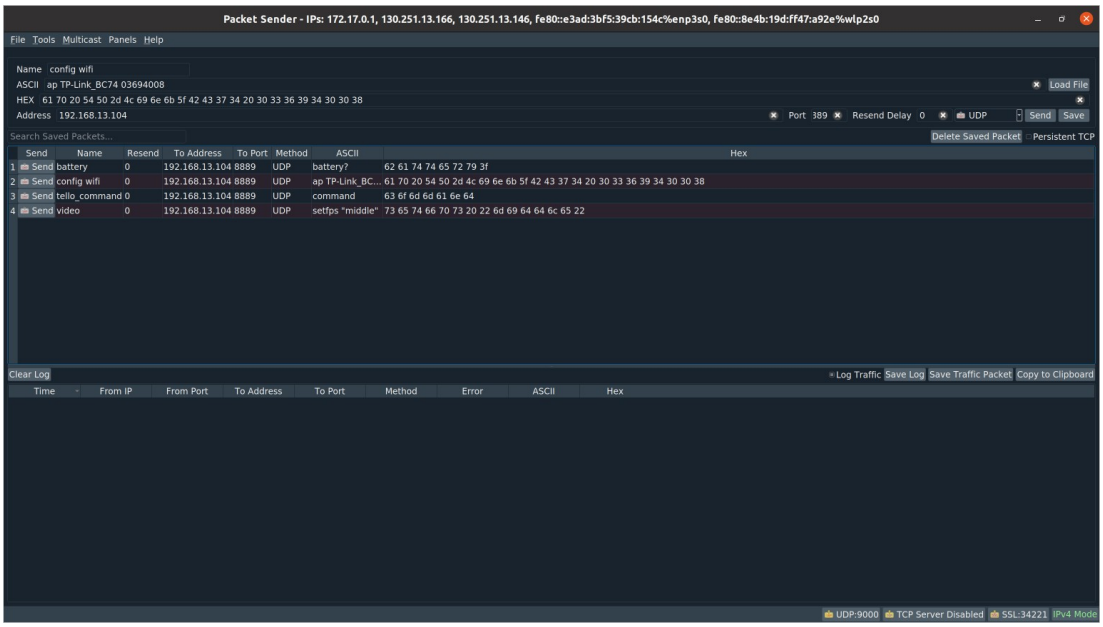

Figure 3.5: Packet sender user interface

Sender application [1] to establish a triangular network connection.

3.5.4 Syncthing application

Syncthing [2] is an open-source application designed for continuous file synchronization across multiple devices.

In a setup involving different computers each with its own operating system, Syncthing can synchronize files across all these entities. Each machine would run an instance of Syncthing, and they would form a peer-to-peer network. As files are changed on any device, Syncthing detects the modifications and propagates them to the other connected devices, ensuring all devices have the latest version of the files, thus facilitating seamless data synchronization in a mixed environment.

3.5.5 SDK (Software Development Kit)

The SDK for Tello EDU includes additional commands and functionalities, such as support for controlling multiple drones simultaneously (drone swarms) and

[1] refer to the address for more information such as downloading the specific system version https://packetsender.com/download

[2] website address to download the application https://syncthing.net/

more sophisticated flight patterns. also Because the SDK is essentially a list of commands and a communication protocol, it's not restricted to any specific programming language. However, popular implementations and libraries have been developed in Python, JavaScript (Node.js), and other languages, which make it easier for developers to get started.

The SDK primarily uses UDP (User Datagram Protocol) as a communication protocol with the drone. There are specific ports assigned for command and control, video streaming, and event logging.some of the SDK modes and commands are described below:

- **Command Mode** Once connected, developers send the "command" command to the drone to enter SDK mode, which allows the sending and receiving of further commands.

- **Command Types** The Tello drone's SDK offers a versatile set of command types that facilitate diverse interactions with the device. Control Commands provide fundamental flight instructions, encompassing actions such as "takeoff", and "land", and directional commands like "up", "down", "left", and "right". On the other hand, Setting Commands enable users to modify configuration settings, including adjustments to the Wi-Fi SSID and password.

  To glean insights about the drone's status, Read Commands can be utilized, which fetch vital information like the drone's current speed, battery level, and the strength of its Wi-Fi signal. Additionally, for those keen on multimedia applications, the SDK includes Video Commands, dedicated to managing and orchestrating video streaming capabilities.

- **Response Types** When a command is sent to the Tello drone, it will return a response. This can be an "OK" if the command was successfully received and executed, or an "ERROR" if there was a problem.

3.5.6 Tello Command types Table

The SDK commands received by Tello can be grouped into three basic types.
Control command (xxx)

- Tello returns "ok" if the command was executed successfully.
- Tello returns an "error" or a result code if the command failed.

Setting command (xx a)

- will attempt to set a new sub-parameter value (a).
- Tello returns "ok" if the command was executed successfully.
- Tello returns an "error" or a result code if the command fails.

Read command (xx?)

- Read the real-time sub-parameter value.

Command	Description	Possible Response
command	Enter SDK command mode	ok/error/inactive
takeoff	Auto take-off	ok/error
land	Auto landing	ok/error
streamon	Turn on the video stream	ok/error
streamoff	Turn off the video stream.	ok/error
forward x	Fly forward by x cm	ok/error + error status
ccw x	rotate counterclockwise by x° ,x=1-360	ok/error + error status

Table 3.2: some UDP Tello control commands

Command	Description	Possible Response
ap ssid pass	Switch Tello to Station mode	ok,drone will reboot in 3s
speedx	Set the current speed to x cm/s	ok/error

Table 3.3: some UDP Tello setting commands

Command	Description	Possible Response
speed?	Get the current set speed (cm/s)	x
battery?	Get the percentage indicating the current battery level	x
ap?	access point name and pass	name and password

Table 3.4: some UDP Tello Read commands

3.6 Environmental setup

In our experimental setup, I used my personal laptop as indicated in table 3.5. Furthermore, to interface all drones with the computer, we utilized the Packet Sender application. By sending the "command" command to the drones, they were prompted to enter SDK mode. Following this, we used "ap SSID pass" command from the tello SDK 3.0 User Guide, to transition Tello into Station mode and connect it to a simple TP-Link router. Each drone, upon selection, reboots within 3 seconds, and this procedure is iterated for all drones to facilitate their connection to the router.

Characteristic	Utility
OS name	Ubuntu 20.04 LTS, Ros Noetic
OS type	64-bit
Manufacturer	Lenovo
Model name	Ideapad L330
RAM	18,9 GiB
CPU	Intel® Core™ i7-8550U CPU @ 1.80GHz × 8
Disk capacity	2.2 TB
GPU	2 GB RADEON

Table 3.5: control system information

Characteristic	Utility
OS name	Ubuntu 20.04 LTS
RAM	3 GB
CPU	Intel® Core™ i7-8550U CPU @ 1.80GHz × 3
Disk capacity	27GB

Table 3.6: Virtual machine information

We established a configuration where each drone was associated with its own virtual machine to handle socket operations internally, find the detailed config-

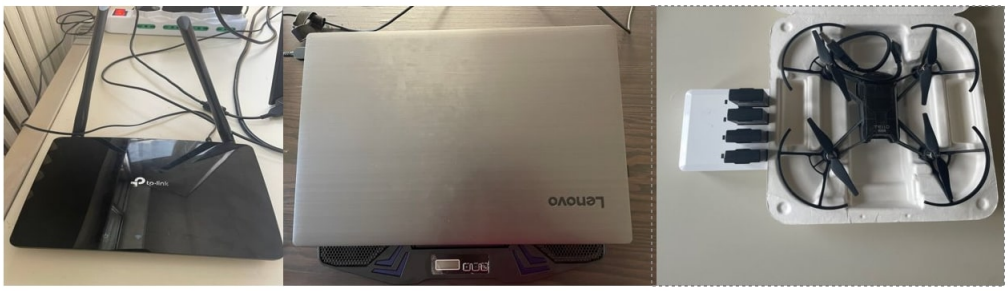

Figure 3.6: complete setup

uration of one VM as an example in table 3.6. Within each virtual machine, we implemented an identical setup to that of the host machine, which was my personal computer. This setup included the installation of necessary libraries and dependencies, such as the Syncthing application for managing network operations.

This step was of utmost importance because we had previously encountered an issue related to the Tello library, particularly concerning synchronized socket binding and release. In our experimental scenario involving multiple drones operating together, all observer drones needed to bind to the same socket simultaneously. However, due to the inherent waiting time associated with socket operations, which exceeded 60 seconds, this configuration introduced significant latency during our tests.

To address this issue and ensure smooth operation within multi-drone formations, we adopted the strategy of creating individual virtual machines for each drone. This approach not only resolved the synchronization problem but also allowed us to maintain consistency in our testing environment by replicating the same setup and dependencies present on the host machine. By doing so, we were able to conduct experiments with reduced latency and more accurate results. This meticulous configuration played a crucial role in enhancing the reliability and precision of our experiments.

3.6.1 communication setup

To facilitate seamless communication among all agent drones, it was imperative to establish a shared environment where each agent could broadcast its information whenever it was assigned a task, whether it was executing the task or verifying its performance. Furthermore, this shared environment allowed agents to access real-time data from other agents, particularly in cases where one agent's information was lost, and others needed to retrieve the latest data.

To achieve this communication framework, we followed the following proce-

dure:

Initialization: After completing the initialization procedure on each machine, we proceeded to create a shared folder named "Sync" on all operating systems. In our test environment, which initially consisted of three virtual boxes and one host machine (with the potential for expansion in the future), this shared folder served as a central hub for data exchange.

- **Data Format**: Each agent had the responsibility of recording its information in a specific format within a .txt file situated in the shared "Sync" folder. This format was designed to provide a structured representation of the agent's status and actions. The information was organized as follows:

  Task Flag (0/1): This flag served as an indicator, with a value of 0 representing an agent responsible for observation and verification tasks, while a value of 1 designated an agent actively engaged in flying tasks.

  Current Angle: This parameter denoted the current orientation angle of the drone as it occupied a specific tile. It offered insight into the drone's heading.

  Current Tile Index: This index represented the number or position of the tile that the drone was currently located on. Tile indexing typically began at 0 and incremented accordingly.

  X and Y Coordinates: These coordinates were calculated using a specific formula, allowing us to pinpoint the drone's precise location within the test environment.

  By adhering to this standardized format, we ensured that data recorded by all agents followed a consistent structure. This uniformity facilitated easy parsing and interpretation of the information, enabling efficient data exchange and coordination among the agents. The task flag, angle, tile index, and coordinates collectively provided comprehensive insights into each agent's role, position, and activities within the multi-agent system.

- **Synchronization**: Every time a small change occurred in one of the machines within the network, the change was automatically propagated to all other machines. This synchronization process ensured that all agents had access to the most up-to-date information from their peers.

By implementing this communication method and establishing a shared folder, we created a robust and efficient means for agents to exchange information, enabling effective coordination and collaboration among them. This approach was pivotal in maintaining real-time awareness and facilitating cooperative decision-making within the multi-agent system.

3.6.2 Live test environment

To conduct the physical tests, we utilized the RICE laboratory. To ensure the safety of everyone involved, strict safety measures were implemented, including the evacuation of other lab members from the testing area. We set up a dedicated station, illustrated in Figure 3.7, to closely monitor the experiments and safely maneuver all the flying drones within a confined space.

With these precautions in place, we were able to collect valuable data and effectively test the networking system and the broader framework by dynamically modifying parameters. For a more comprehensive understanding of the testing methodology and the results obtained, please refer to Chapter 4. This chapter provides additional insights into the specifics of our testing procedures and the outcomes we achieved.

Figure 3.7: environmental setup

3.7 system architecture

The system's architecture in figure 3.7.3 is broken down into three primary components:

3.7.1 ROS-based Trust Framework

This Framework, with its methodical design and well-defined packages, ensures that tasks are appropriately assigned, trust is maintained, and the system op-

erates cohesively. For more in-depth information about this framework, refer to 3.1.3.

3.7.2 Performing package

Another vital component within our system architecture is the "performing package." This package houses a variety of algorithms, each contributing to the overall functionality of the navigation system. One such algorithm is the "map-graph-generator," which plays a pivotal role in creating a graph representation of the environment. Additionally, the A* path planning algorithm" is employed to determine the most efficient route through obstacles to reach a specified destination and then come back to the initial position with the return-to-home function. Furthermore, this package includes a crucial function responsible for activating the Tello drone's SDK mode, subsequently issuing precise commands to guide the drone along the predefined path. Importantly, it also calculates any necessary rotation adjustments required for seamless transitions between tiles, ensuring the drone's precise movement.

3.7.3 observation package

On the other hand, the "observation component" encompasses various packages geared toward enhancing our system's perception and situational awareness capabilities:

- **Camera Calibration**: To calibrate the cameras on our Tello drones, we implemented a meticulous process. We utilized a 0.03-meter square calibration checkerboard, capturing approximately 50 images from various angles along each axis (x, y, z) for each individual Tello drone in our fleet. To streamline the image acquisition process from the Tello cameras, we developed a Python script named "data-generation-tello-camera."

  This Python code is designed to interface with the Tello drones, establish a connection to the camera through activating SDK mode, and initiate a video stream. It performs several essential tasks to facilitate the calibration process. First, it retrieves and displays the Tello's battery level, ensuring the drone's readiness for the operation. Additionally, the script sets up a counter to keep track of the images captured during the calibration process.

  The key function within the script is "take-picture()," which is responsible for capturing frames from Tello's camera and saving them as individual images. In an infinite loop, the script continuously reads frames from the camera, displays the live feed using the OpenCV library, and waits for user input. When the user presses the "c" key, the "take-picture()" function is

triggered to capture an image. It then increments the counter and saves the image with a corresponding filename. This loop continues until the user decides to exit by pressing the "ESC" key, at which point the program gracefully terminates.

After repeating these steps for all the Tello drones in our fleet, we aggregated the paths to the saved images. Subsequently, we executed the "calibration.py" file, which facilitated the retrieval of intrinsic parameters and distortion coefficients. This meticulous calibration process ensures that the Tello drone cameras provide accurate and undistorted images, laying the foundation for precise and reliable visual data acquisition in various operational scenarios.

Figure 3.8: calibration checkerboard

- **ArUco Pose Estimation**: To perform camera pose estimation using ArUco markers, we followed a systematic approach that involved several steps:

  ArUco Tag Generation: We initiated the process by generating ArUco tags of the preferred size and ID using the online tool available at [1] These tags would serve as reference markers for the camera pose estimation. We used the "ArUCo-Markers-Pose-Estimation-Generation-Python" GitHub repository, which contains all the necessary code for detecting ArUco tags in both images and videos. This repository enabled us to not only detect the tags but also estimate the pose of the object relative to the camera. It provided comprehensive information and resources for camera pose estimation.

[1] refer to this address for more information, "https://chev.me/arucogen/"

the camera calibration step generated two ".npy" files, namely "calibration-matrix.npy" and "distortion-coefficients.npy." These files store critical information about the camera's intrinsic parameters and distortion coefficients, respectively. It's worth noting that this calibration process did not require the Tello drone to be connected to Wi-Fi or any network. With the calibration data in hand, we connected our Tello drone to the network and executed the "aruco-detection-live.py" script. This script allowed us to detect and track the ArUco markers in real time using the Tello's onboard camera. To enable accurate camera pose estimation, we integrated the calibration data into the "aruco-pose-estimation.py" script. Specifically, we inserted the values from the "calibration-matrix.npy" and "distortion-coefficients.npy" files into the "intrinsic-camera" and "distortion" variables, respectively. After integrating the calibration data, we reconnected our Tello drone and reran the "aruco-detection-live.py" script. This time, the script utilized the calibrated camera parameters to improve the accuracy of marker detection and camera pose estimation.

By following these steps, we achieved reliable camera pose estimation using ArUco markers with our Tello drone, ensuring that we could accurately determine the position and orientation of objects within the drone's field of view.

- **Drone Localization**: Building upon the data gathered from the previous components, the "drone localization" package takes the localization process to a more comprehensive level. It extends beyond solely relying on ArUco markers and works to localize the drones within the broader context of a global reference frame. Within this package, several functions come into play. Initially, they compute the camera's position in relation to the global frame within the given environment. Subsequently, they estimate and calculate the precise position of the drone agent (often referred to as the "winner") within the world frame. This comprehensive localization capability ensures that the drones can navigate and operate effectively within a larger spatial context.

In this schematic overview of the system architecture, the ROS (Robot Operating System) framework initiates a TCP connection to each adapter, located inside individual virtual boxes, through a socket binding on their unique IP and port combinations. Once these connections are successfully established, the framework proceeds to transmit a string message to each adapter. As an example, the message might be in the format "G1—PERFORMING—E1/A1/0" or "G1—VERIFYING—E1/A1/0" where E stands for event and A stands for action. Importantly, only one adapter will receive the "PERFORMING" string,

while the others will receive the "VERIFYING" string.

Each adapter, upon receiving the message, performs some modifications on the initial string to extract the command part. Based on this extracted command, the corresponding function is then triggered to proceed accordingly, whether it involves performing a task or verifying certain parameters. Subsequently, the command is sent through a UDP socket connection to each drone, utilizing their respective IP addresses and ports as illustrated in the figure. This architecture facilitates effective communication and coordination between the ROS framework, the adapters, and the drones, enabling the system to execute tasks and verifications seamlessly.

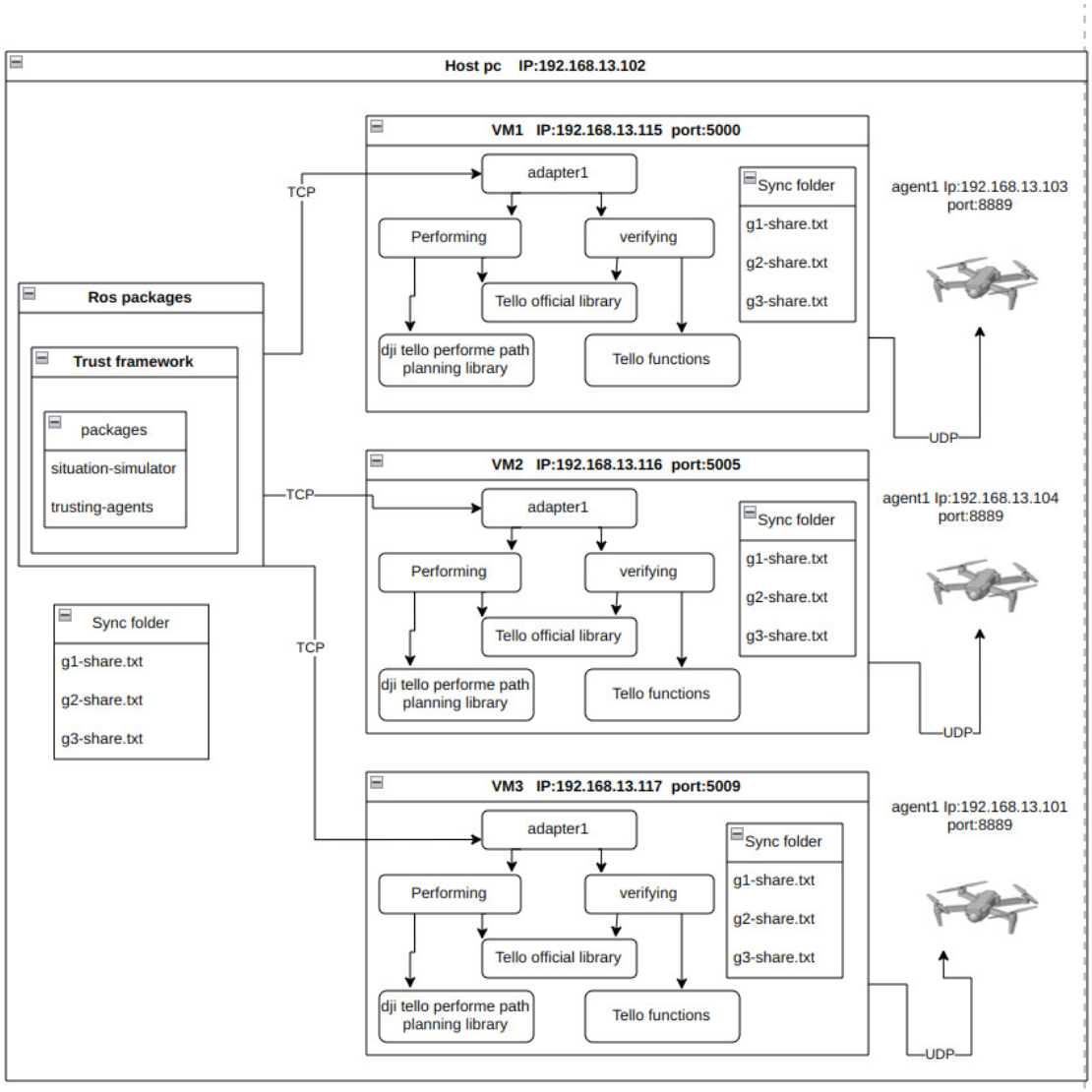

Figure 3.9: system software architecture

Chapter 4

test results

4.1 initial stage

In the initial testing stage, our primary objective was to validate the accuracy of the drone's localization capabilities. We achieved this by employing Aruco marker pose estimation, followed by rigorous calculations. The precision of our estimation is depicted in Figure 4.1, where we observed an offset falling within the range of ±3 cm in at least 60 tests on real robots. This level of precision was achieved despite the inherent challenges in estimating the drone's position in a dynamic environment.

Expected	Real
x=51 ,y=65	x=53.2 ,y=65.1
x=120 ,y=0	x=122.6 ,y=2.9
x=90 ,y=150	x=89.1 ,y=148.5
x=60 ,y=60	x=58.9 ,y=58.1
x=120 ,y=60	x=122.1 ,y=61.9
x=0 ,y=180	x=2.1 ,y=181.9
x=60 ,y=180	x=58.4 ,y=178.2
x=0 ,y=60	x=2.2 ,y=62.1
x=180,y=120	x=178.5 ,y=119.1

Table 4.1: data offset comparison in theory and real experiment

In order to compute the standard deviation, we performed a statistical analysis on a specific dataset. This sample dataset was comprised of nine pairs of x and y coordinates, which were conveniently provided in the table located above our analysis. These pairs of values were crucial for our calculations and played a fundamental role in determining the variability and dispersion of the data.

Calculate Differences in x-values:

$$dx_i = x_{\text{real},i} - x_{\text{expected},i}$$

Similarly, for the y-values:

$$dy_i = y_{\text{real},i} - y_{\text{expected},i}$$

Calculate the Mean of the Differences for the x-values:

$$\bar{dx} = \frac{\sum dx_i}{n} = 0.67$$

For the y-values:

$$\bar{dy} = \frac{\sum dy_i}{n} = -0.57$$

Calculate the Standard Deviation of the Differences for the x-values:

$$s_{dx} = \sqrt{\frac{\sum (dx_i - \bar{dx})^2}{n-1}} = 1.7860$$

And for the y-values:

$$s_{dy} = \sqrt{\frac{\sum (dy_i - \bar{dy})^2}{n-1}} = 1.9730$$

Upon analyzing the discrepancies between the theoretical and actual data sets, the calculated standard deviations for the x and y values were found to be 1.7860 and 1.9730 respectively. These values represent the average deviations of the data points from their respective means. Given the dimensions of each Tello drone being 10cm x 10 cm x 5 cm and considering the environmental unit size of 60cm (with each tile measuring 60cm x 60cm), such discrepancies in data can be overlooked.

This suggests that while there are variations between the theoretical predictions and the actual measurements, the magnitude of these variations is consistent and within expected boundaries. The data's spread, as captured by the standard deviations, is in alignment with the thresholds set for this investigation, indicating that the real data and theoretical model are reasonably congruent.

4.1.1 Exploring Factors Behind Data Discrepancies

Numerous elements contribute to the disparity between the expected and observed data. Among these factors, hardware limitations frequently come into

Figure 4.1: camera pose estimation values in real experiments, from left to right the expected values were x=120,y=0; x=51,y=65; x=31, y=-30.

play. To illustrate, as soon as the drone's battery level falls below the 40 percent threshold, it triggers a rise in temperature within the device. This temperature increase, specifically on the motherboard, frequently leads to noticeable complications with the camera's operation. Consequently, it becomes evident that the intrinsic hardware constraints within our system, such as the drone's response to low battery levels or prolonged camera usage (beyond 10 minutes), exert a substantial influence in elucidating the observed deviation between projected and actual data.

4.2 second phase

The second phase of testing was dedicated to verifying network connectivity and ensuring that communication between all the drones was established within the local network. To accomplish this, we followed a specific sequence of steps. Initially, we connected all the drones to a router, and then we connected the router to the host computer, which happened to be my personal computer. The host computer, in turn, was linked to the laboratory's Wi-Fi network via a LAN cable.

In this configuration, each drone possessed its own unique IP address, but they all shared the same UDP port (8889) for activating the drone control mode [1]. Once the TCP connection between the trust framework and adapters was successfully established, the winning drone initiated its flight by binding to a socket on the local host IP address and port 8889. The flights were executed

[1] for more details, please refer to the SDK website https://dl.djicdn.com/downloads/RoboMaster+TT

flawlessly in all of our tests. However, upon completing their missions, the drones released their respective sockets.

A challenge emerged during this process, primarily related to the waiting time associated with each socket. In most cases, this waiting period exceeded 60 seconds, resulting in a noticeable delay within the system. In other words, even if a socket was available for binding, it couldn't be utilized immediately due to this extended wait time. To address this issue, we made certain modifications to the Tello official library, allowing the socket to be reused if it was within its designated waiting time. Despite these improvements, all the observer drones still encountered difficulty in simultaneously performing the verification tasks, which was contrary to our initial objective. To resolve this challenge, we devised a solution that incorporated the use of the Syncthing application, as mentioned in 3.5.4. Additionally, we employed Oracle VirtualBox to create a separate virtual machine for each drone. These virtual machines were instrumental in managing socket binding operations locally, further optimizing our system's performance and addressing the socket waiting time issue.

4.3 final phase

Afterward, we advanced to the final testing phase, where our primary goal was to evaluate how well all system components, including the reliability and Trust updates in the framework, integrated and interacted seamlessly.

4.3.1 Experiment results, one perfect agent,two with 60% and 70% rate of observing correctly

Before explaining the experiment details let's understand the values of the table 4.2. There are 3 sets of values in each cell, the first value indicates the success rate in performing an action, and the second and third numbers are the optimism and pessimism rates for observing the action, (TP) and (TN) respectively. For example, G3 with these values 1.00;0.6;1.00 (toward itself) holds strong confidence in its ability to perform an action (number 1.00) while having only a 60% optimism rate when assessing the outcome of actions it observes. The last number 1.00, indicates 100% pessimism when assessing the outcome of actions. This means the agent is always expecting negative outcomes or true negatives. In this scenario, all three agents have performed a single action. Remarkably, G1 and G2 deemed their respective actions as successful (for action result verifications for G1 and G2, please refer to tables 4.7 and 4.6, respectively). On the contrary, G3 regarded their action as a failure (for the result verification table for G3, please see 4.8). Meanwhile, the other two observing agents, in the first case, G2 and G3, and in

the second case, G1 and G3, and in the last case, G1 and G2 perceived these actions as failures.

The perceived reliability of G1 for action 2 and G2 for action 1 is recorded as 1.00, indicating their strong belief in the success of their actions, even though their observations contradict this belief. Their verification trustworthiness(refer to 4.3.2 for more details on trust metrics interpretation) however, is low (-0.33), as their observations clash with the accounts of the other two agents. On the other hand, G3 executed action 3 but evaluated it as unsuccessful, with a perceived reliability of 0.00. Interestingly, in this case, G3's observation aligns with the accounts of the other two agents, resulting in all three agents having a verification trustworthiness score of 1.0 for action 3 (although G3 failed to perform or does not believe in themselves, the trust increased in values, updating a higher trust in the system).

The primary factor contributing to the negative and low reliability in the system can be attributed to the inherent limitations of the agents in effectively observing and verifying the outcomes of the performing drone, camera initialization failure or onboard Aruco markers were not visible due to the performer drone's landing angle. These limitations in their ability to gather accurate and consistent information regarding the drone's actions and results have a direct impact on their trust in the system and each other. Inadequate observation and verification mechanisms can lead to discrepancies in the agents' assessments and, consequently, result in lower levels of perceived reliability. Addressing these limitations and improving observation and verification processes is crucial for enhancing the overall trustworthiness and performance of the system.

	G1	G2	G3
G_1	1.00;1.00;1.00	1.00;1.00;1.00	1.00;1.00;1.00
G_2	1.00;0.7;1.00	1.00;0.7;1.00	1.00;0.7;1.00
G_3	1.00;0.6;1.00	1.00;0.6;1.00	1.00;0.6;1.00

Table 4.2: Self-declaration and assumptions of each agent towards all the other agents

4.3.2 Trust metrics update

In order to create a general idea of how these trust metrics are updated in the framework we will go through the details of the table 4.3 as an example. In the following tables for G2 4.5, and G3 4.4, you can find the ultimate trust updates for each agent concerning their perceptions of one another. In these tables, the maximum level of trust is 1 and the minimum is -1. This detailed analysis helps to understand how G1 evaluates the abilities and observations of the other agents,

as well as the execution and verification of actions in the given scenarios (the pattern is the same for G2 and G3).

In table 4.3, the first column pertains to G1's assessments of the performance and reliability of agents G1 (itself), G2, and G3 in executing action A1. Given that only G2 has attempted A1 so far and it resulted in failure, all the trust values in this column are set to 0, indicating low trust in their ability to perform A1 successfully.

The second column addresses the agents' proficiency in observing the outcomes of action A1. In this case, because G1 and G3 concurred in their observations, while G2's observations differed, G1 and G3 exhibited higher trust values compared to G2. This discrepancy reflects the agreement between G1 and G3, leading to a greater level of trust in their observations.

Continuing further, the third column of the table is dedicated to the assessment of how G1 perceives G1, G2, and G3's reliability and ability to perform action A2. In this case, G1 holds itself in high regard when it comes to executing action A2, assigning a trust value of 1 to itself (since G1 won the task and executed action A2), indicating absolute self-confidence and competence. while for others this value is zero.

Now, moving to the fourth column, which relates to the verification phase, since G2 and G3 reached an agreement on their observations, both perceived the result as false, while G1 disagreed with their observations. Consequently, the values for G2 and G3 are less than the values assigned to G1 in the verification trust column.

The fifth and sixth columns in the table are associated with the execution and verification of action A3, seen from G1's perspective.

In the fifth column, G1's trust levels are notably low towards all entities, including itself, when it comes to executing action A3. This low trust can be attributed to the fact that G3 attempted action A3 and experienced a failure, as indicated in reference 4.8. This unfavorable outcome likely contributed to G1's skepticism and low trust values assigned to all agents for this specific action.

In contrast, the last column, representing the verification of the execution, shows all values set at 1.00. This signifies unanimous agreement among all agents regarding the observation results. Regardless of the initial low trust in executing action A3, the agents seem to have reached a consensus when it comes to verifying the outcomes, as all values are harmonized at 1.00.

G_1	0.00	0.33	1.00	-0.33	0.00	1.00
G_2	0.00	-0.33	0.00	0.33	0.00	1.00
G_3	0.00	0.33	0.00	0.33	0.00	1.00

Table 4.3: Trust metrics at the end of the experiment according to G1

G_1	0.00	0.33	0.00	-0.33	0.00	1.00
G_2	0.00	-0.33	0.00	0.33	0.00	1.00
G_3	0.00	0.33	0.00	0.33	0.00	1.00

Table 4.4: Trust metrics at the end of the experiment according to G3

G_1	0.00	0.33	0.00	-0.33	0.00	1.00
G_2	1.00	-0.33	0.00	0.33	0.00	1.00
G_3	0.00	0.33	0.00	0.33	0.00	1.00

Table 4.5: Trust metrics at the end of the experiment according to G2

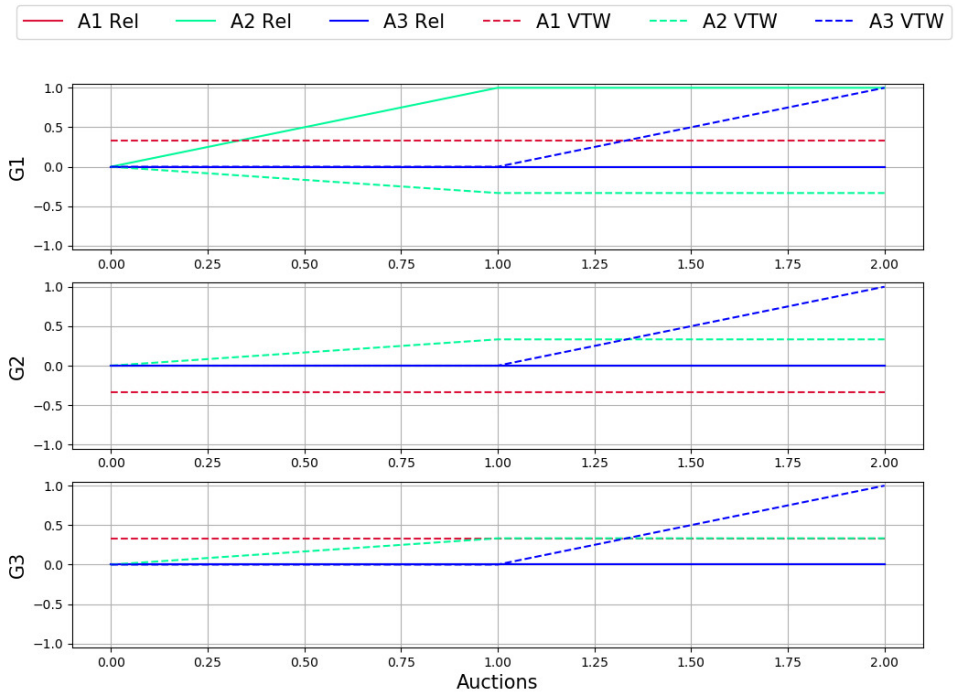

Figure 4.2: Trust dynamics of the experiment according to G1

Auction ID	Declarant	Performer	Perceived Result
E1/A1/0	G2	G2	True
E1/A1/0	G3	G2	False
E1/A1/0	G1	G2	False

Table 4.6: Result verification for E1/A1/0 , G2 performed

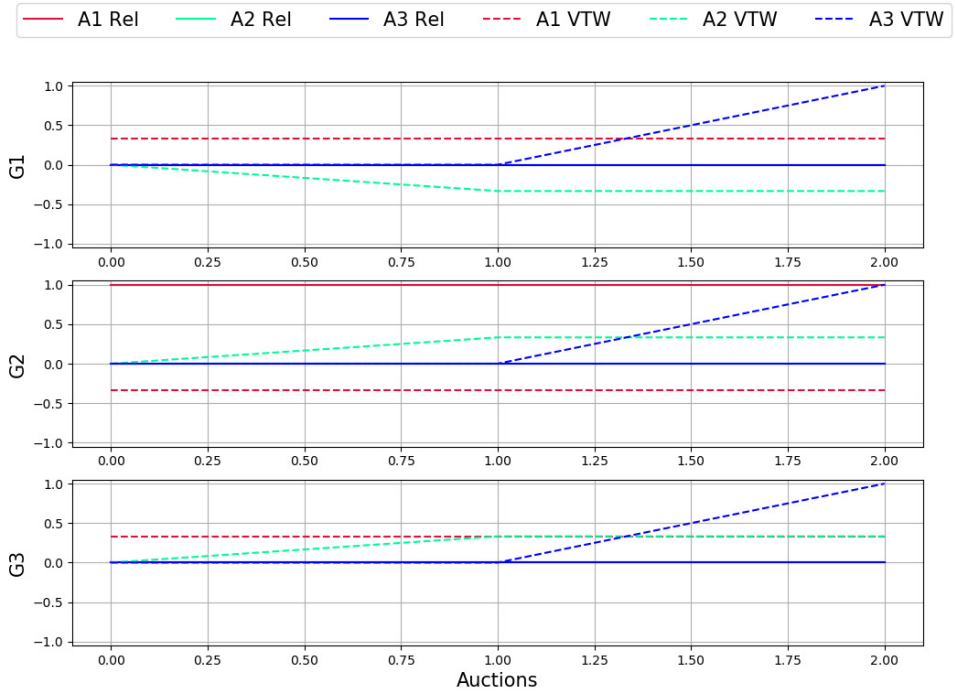

Figure 4.3: Trust dynamics of the experiment according to G2

Auction ID	Declarant	Performer	Perceived Result
E2/A2/1	G1	G1	True
E2/A2/1	G3	G1	False
E2/A2/1	G2	G1	False

Table 4.7: Result verification for E2/A2/1 , G1 performed

Auction ID	Declarant	Performer	Perceived Result
E3/A3/2	G1	G3	False
E3/A3/2	G3	G3	False
E3/A3/2	G2	G3	False

Table 4.8: Result verification for E3/A3/2, G3 performed

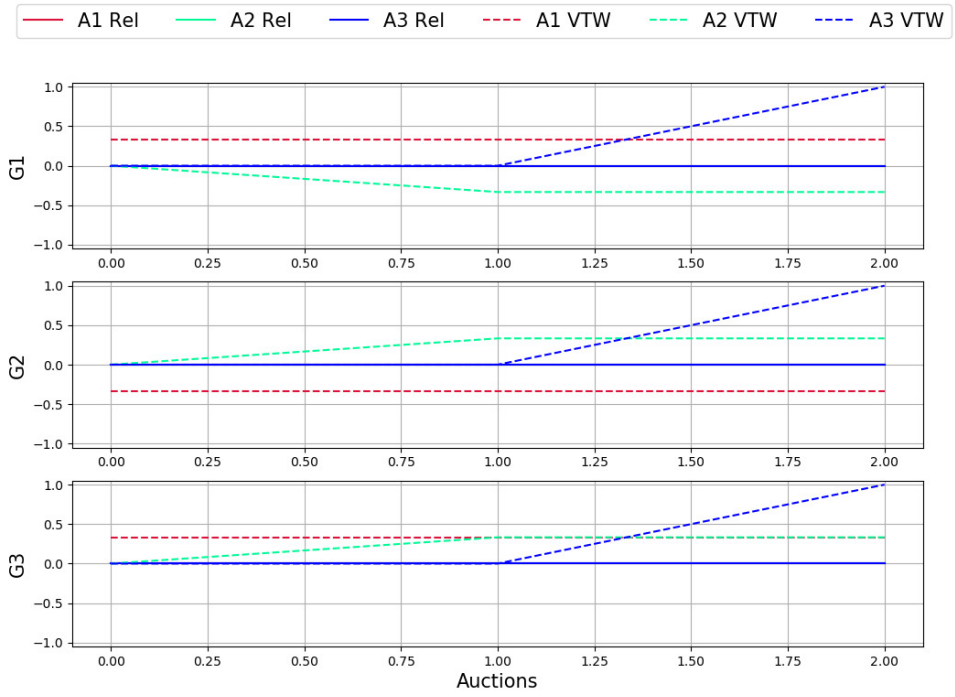

Figure 4.4: Trust dynamics of the experiment according to G3

Chapter 5

conclusion

5.1 Recap

In this research, the primary goal was to assess how well the Trust framework could be implemented and function effectively in the context of UAVs. It is worth noting that this framework had previously been examined and tested in the domain of humanoid robots. The thesis aimed to expand the scope of its application by exploring its effectiveness when dealing with UAVs, which are fundamentally different in terms of functionality, operation, and the trust dynamics involved. This shift in focus from humanoid robots to UAVs is a significant aspect of this research, aiming to provide insights into the framework's adaptability across diverse technological platforms.

Furthermore, in this work Trust is emphasized as a crucial factor when multiple robots from different companies collaborate, highlighting the need for a robust system for communication and cooperation among robots. It also introduces the integration of Aruco markers for precise localization. Additionally, it mentions the importance of synchronized control of multiple drones and the use of different networking approaches for establishing communication among them.

5.2 System strengths

Three advantages of the mentioned system are as below:

- **Enhanced Cooperation**: The trust-centric framework for task assignment fosters cooperation among robotic agents, enabling them to exchange information, assign tasks, and adapt to changing conditions. This leads to improved operational outcomes and mission fulfillment.

- **Precise Localization**: The integration of Aruco markers provides UAVs with reliable and precise positioning and orientation information, even in scenarios with GNSS signal limitations or occlusions. This ensures cohesive inter-robot operations and accurate camera pose identification.
- **Scalability and Ease of Deployment**: The use of TCP and UDP protocols for communication between drone agents emphasizes ease of deployment and scalability for future applications. This approach facilitates efficient data exchange among drones and supports real-time communication for controlling multiple UAVs.

5.3 Future work

The derived system architecture demonstrates its versatility on UAVs performing a wide spectrum of tasks. This adaptability is a key advantage, as it enables researchers and practitioners to apply the trust-centric framework to different UAV models and mission profiles. This work paves the way for future examinations by providing a flexible foundation for assessing the system's performance in real-world scenarios. Moreover, it encourages the exploration of new applications and optimizations, making it a valuable resource for advancing the field of collaborative robotics and autonomous systems.

References

[1] Aggarwal, S. & Kumar, N. (2020). Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, **149**, 270–299. 14

[2] Alcántara, A., Capitán, J., Torres-González, A., Cunha, R. & Ollero, A. (2020). Autonomous execution of cinematographic shots with multiple drones. *IEEE Access*, **8**, 201300–201316. 6

[3] Alsadik, B. & Karam, S. (2021). The simultaneous localization and mapping (slam)-an overview. *Surv. Geospat. Eng. J*, **2**, 34–45. 12

[4] Bradski, G. & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.". 17

[5] Budiharto, W., Gunawan, A.A., Suroso, J.S., Chowanda, A., Patrik, A. & Utama, G. (2018). Fast object detection for quadcopter drone using deep learning. In *2018 3rd international conference on computer and communication systems (ICCCS)*, 192–195, IEEE. 10

[6] Calvaresi, D., Mualla, Y., Najjar, A., Galland, S. & Schumacher, M. (2019). Explainable multi-agent systems through blockchain technology. In *Explainable, Transparent Autonomous Agents and Multi-Agent Systems: First International Workshop, EXTRAAMAS 2019, Montreal, QC, Canada, May 13–14, 2019, Revised Selected Papers 1*, 41–58, Springer. 3

[7] Carabelea, C., Boissier, O. & Florea, A. (2003). Autonomy in multi-agent systems: A classification attempt. In *International Workshop on Computational Autonomy*, 103–113, Springer. 5

[8] Carmigniani, J. & Furht, B. (2011). Augmented reality: an overview. *Handbook of augmented reality*, 3–46. 16

[9] Catterson, V.M., Davidson, E.M. & McArthur, S.D. (2012). Practical applications of multi-agent systems in electric power systems. *European Transactions on Electrical Power*, **22**, 235–252. 6

[10] Chakraborty, S. & Gupta, S. (2014). Medical application using multi agent system-a literature survey. *International Journal of Engineering Research and Applications*, **4**, 528–546. 6

[11] Chenjie, W., Bin, L., Chengyuan, L., Wei, W., Lu, Y. & Qing, Z. (2020). The collaborative mapping and navigation based on visual slam in uav platform. *Acta Geodaetica et Cartographica Sinica*, **49**, 767. 13

[12] Cho, J.H., Chan, K. & Adali, S. (2015). A survey on trust modeling. *ACM Computing Surveys (CSUR)*, **48**, 1–40. 21

[13] Choi, H.L., Brunet, L. & How, J.P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, **25**, 912–926. 1

[14] Cyba, A., Szolc, H. & Kryjak, T. (2021). A simple vision-based navigation and control strategy for autonomous drone racing. In *2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR)*, 185–190, IEEE. 2

[15] Cyganek, B. & Gruszczyński, S. (2014). Hybrid computer vision system for drivers' eye recognition and fatigue monitoring. *Neurocomputing*, **126**, 78–94. 16

[16] De Corso, T., De Vito, L., Picariello, F., Wojtowicz, K., Marut, A. & Wojciechowski, P. (2023). Optical multi-camera uav positioning system via aruco fiducial markers. In *2023 IEEE 10th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, 352–357. 13

[17] Desai, M., Stubbs, K., Steinfeld, A. & Yanco, H. (2009). Creating trustworthy robots: Lessons and inspirations from automated systems. 1

[18] Elmeseiry, N., Alshaer, N. & Ismail, T. (2021). A detailed survey and future directions of unmanned aerial vehicles (uavs) with potential applications. *Aerospace*, **8**, 363. 8

[19] Famili, A. & Park, J.M.J. (2020). Rolatin: Robust localization and tracking for indoor navigation of drones. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 1–6. 12

[20] Famili, A., Stavrou, A., Wang, H. & Park, J.M. (2023). idrop: Robust localization for indoor navigation of drones with optimized beacon placement. *IEEE Internet of Things Journal*, **10**, 14226–14238. 12

[21] Famili, A., Stavrou, A., Wang, H. & Park, J.M. (2023). Pilot: High-precision indoor localization for autonomous drones. *IEEE Transactions on Vehicular Technology*, **72**, 6445–6459. 12

[22] Fang, R. & Cai, C. (2021). Computer vision based obstacle detection and target tracking for autonomous vehicles. In *MATEC Web of Conferences*, vol. 336, 07004, EDP Sciences. 15

[23] Ferguson, D., Likhachev, M. & Stentz, A. (2005). A guide to heuristic-based path planning. In *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, 9–18. 14

[24] Gao, Z., Wanyama, T., Singh, I., Gadhrri, A. & Schmidt, R. (2020). From industry 4.0 to robotics 4.0-a conceptual framework for collaborative and intelligent robotic systems. *Procedia manufacturing*, **46**, 591–599. 2

[25] Gaol, J.L., Haq, M.A., Armiyanto, S., Kusuma, H. & Tasripan, T. (2019). Automatic self-checkout system using surf, brute force matcher, and rfid for payment process optimization at supermarket. *IPTEK Journal of Proceedings Series*, 140–144. 17

[26] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F.J. & Marín-Jiménez, M.J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, **47**, 2280–2292. 3

[27] Gasparetto, A., Boscariol, P., Lanzutti, A. & Vidoni, R. (2015). *Path Planning and Trajectory Planning Algorithms: A General Overview*, 3–27. Springer International Publishing, Cham. 14

[28] Gatteschi, V., Lamberti, F., Paravati, G., Sanna, A., Demartini, C., Lisanti, A. & Venezia, G. (2015). New frontiers of delivery services using drones: A prototype system exploiting a quadcopter for autonomous drug shipments. In *2015 IEEE 39th annual computer software and applications conference*, vol. 2, 920–927, IEEE. 11

[29] Ghasemi, M., Varshosaz, M., Pirasteh, S. & Shamsipour, G. (2021). Optimizing sector ring histogram of oriented gradients for human

injured detection from drone images. *Geomatics, Natural Hazards and Risk*, **12**, 581–604. 24

[30] Goel, S., Gabela, J., Kealy, A. & Retscher, G. (2018). An indoor outdoor cooperative localization framework for uavs. In *Proceedings of the International Global Navigation Satellite Systems (IGNSS) Conference*. 12

[31] Granatyr, J., Botelho, V., Lessing, O.R., Scalabrin, E.E., Barthès, J.P. & Enembreck, F. (2015). Trust and reputation models for multiagent systems. *ACM Computing Surveys (CSUR)*, **48**, 1–42. 3

[32] Grillo, A., Carpin, S., Recchiuto, C.T. & Sgorbissa, A. (2022). Trust as a metric for auction-based task assignment in a cooperative team of robots with heterogeneous capabilities. *Robotics and Autonomous Systems*, **157**, 104266. 2, 3, 21, 22

[33] Groom, V. & Nass, C. (2007). Can robots be teammates?: Benchmarks in human–robot teams. *Interaction studies*, **8**, 483–500. 7

[34] He, T., Zeng, Y. & Hu, Z. (2019). Research of multi-rotor uavs detailed autonomous inspection technology of transmission lines based on route planning. *IEEE Access*, **7**, 114955–114965. 10

[35] Hoeing, M., Dasgupta, P., Petrov, P. & O'Hara, S. (2007). Auction-based multi-robot task allocation in comstar. In *Proceedings of the 6th international joint conference on autonomous agents and multiagent systems*, 1–8. 8

[36] Hong, C. & Shi, D. (2018). A cloud-based control system architecture for multi-uav. In *Proceedings of the 3rd International Conference on Robotics, Control and Automation*, 25–30. 3, 14

[37] Huang, S., Sun, G. & Li, M. (2021). Fast and flann for feature matching based on surf. In *2021 33rd Chinese Control and Decision Conference (CCDC)*, 1584–1589, IEEE. 17

[38] Kalaitzakis, M., Carroll, S., Ambrosi, A., Whitehead, C. & Vitzilaios, N. (2020). Experimental comparison of fiducial markers for pose estimation. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 781–789. 13

[39] Kalaitzakis, M., Cain, B., Carroll, S., Ambrosi, A., Whitehead, C. & Vitzilaios, N. (2021). Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag,

aruco and stag markers. *Journal of Intelligent & Robotic Systems*, **101**, 1–26. 3

[40] Kan, T.W., Teng, C.H. & Chou, W.S. (2009). Applying qr code in augmented reality applications. In *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry*, 253–257. 16

[41] Kawecki, A., Dabrowski, P., Januszko, S., Rećko, M. & Dzierżek, K. (2022). Ar tags based absolute positioning system. In *2022 8th International Conference on Automation, Robotics and Applications (ICARA)*, 62–67, IEEE. 16

[42] Khamis, A., Hussein, A. & Elmogy, A. (2015). Multi-robot task allocation: A review of the state-of-the-art. *Cooperative robots and sensor networks 2015*, 31–51. 8

[43] Leventić, H., Keser, T. & Vdovjak, K. (2018). A fast one-pixel wide contour detection method for shapes contour traversal in binary images. In *2018 International Conference on Smart Systems and Technologies (SST)*, 11–14, IEEE. 17

[44] Li, D., Rau, P.P. & Li, Y. (2010). A cross-cultural study: Effect of robot appearance and task. *International Journal of Social Robotics*, **2**, 175–186. 1

[45] Li, G., Li, M., Mao, J., Chen, H. & Li, Y. (2023). A remote multi-uav control system based on smart device. In W. Fu, M. Gu & Y. Niu, eds., *Proceedings of 2022 International Conference on Autonomous Unmanned Systems (ICAUS 2022)*, 135–142, Springer Nature Singapore, Singapore. 4, 14

[46] Liu, L.s., Lin, J.f., Yao, J.x., He, D.w., Zheng, J.s., Huang, J. & Shi, P. (2021). Path planning for smart car based on dijkstra algorithm and dynamic window approach. *Wireless Communications and Mobile Computing*, **2021**, 1–12. 14

[47] Loh, Y.P., Liang, X. & Chan, C.S. (2019). Low-light image enhancement using gaussian process for features retrieval. *Signal Processing: Image Communication*, **74**, 175–190. 17

[48] Lu, Y. & Young, S. (2020). A survey of public datasets for computer vision tasks in precision agriculture. *Computers and Electronics in Agriculture*, **178**, 105760. 15

[49] Lu, Y., Xue, Z., Xia, G.S. & Zhang, L. (2018). A survey on vision-based uav navigation. *Geo-spatial information science*, **21**, 21–32. 12

[50] Merabet, G.H., Essaaidi, M., Talei, H., Abid, M.R., Khalil, N., Madkour, M. & Benhaddou, D. (2014). Applications of multi-agent systems in smart grids: A survey. In *2014 International conference on multimedia computing and systems (ICMCS)*, 1088–1094, IEEE. 6

[51] Mistry, D. & Banerjee, A. (2017). Comparison of feature detection and matching approaches: Sift and surf. *GRD Journals-Global Research and Development Journal for Engineering*, **2**, 7–13. 17

[52] Mohsan, S.A.H., Khan, M.A., Noor, F., Ullah, I. & Alsharif, M.H. (2022). Towards the unmanned aerial vehicles (uavs): A comprehensive review. *Drones*, **6**, 147. 8

[53] Moreno, A. (2003). Medical applications of multi-agent systems. *Computer Science and Mathematics Department, University of Rovira, Spain*. 6

[54] Mostafa, S.A., Darman, R., Khaleefah, S.H., Mustapha, A., Abdullah, N. & Hafit, H. (2019). A general framework for formulating adjustable autonomy of multi-agent systems by fuzzy logic. In *Agents and Multi-Agent Systems: Technologies and Applications 2018: Proceedings of the 12th International Conference on Agents and Multi-Agent Systems: Technologies and Applications (KES-AMSTA-18) 12*, 23–33, Springer. 5

[55] Nahangi, M., Heins, A., McCabe, B. & Schoellig, A. (2018). Automated localization of uavs in gps-denied indoor construction environments using fiducial markers. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, vol. 35, 1–7, IaaRC publications. 3, 13

[56] Nakhaeinia, D., Tang, S.H., Noor, S.M. & Motlagh, O. (2011). A review of control architectures for autonomous navigation of mobile robots. *International Journal of the Physical Sciences*, **6**, 169–174. 11

[57] Neal, K.L. (2014). 5 - be credible: why should you believe me? In K.L. Neal, ed., *Six Key Communication Skills for Records and Information Managers*, 67–89, Chandos Publishing. 22

[58] Panagiotou, P. & Yakinthos, K. (2020). Aerodynamic efficiency and performance enhancement of fixed-wing uavs. *Aerospace Science and Technology*, **99**, 105575. 9

[59] Patrick, O.O., Nnadi, E.O. & Ajaelu, H.C. (2020). Effective use of quadcopter drones for safety and security monitoring in a building construction sites: Case study enugu metropolis nigeria. *Journal of Engineering and Technology Research*, **12**, 37–46. 11

[60] Pedersen, M.R., Nalpantidis, L., Andersen, R.S., Schou, C., Bøgh, S., Krüger, V. & Madsen, O. (2016). Robot skills for manufacturing: From concept to industrial deployment. *Robotics and Computer-Integrated Manufacturing*, **37**, 282–291. 2

[61] Quigley, M., Gerkey, B. & Smart, W.D. (2015). *Programming Robots with ROS: a practical introduction to the Robot Operating System*. " O'Reilly Media, Inc.". 33

[62] Raza, W., Osman, A., Ferrini, F. & Natale, F.D. (2021). Energy-efficient inference on the edge exploiting tinyml capabilities for uavs. *Drones*, **5**, 127. 24

[63] Rezaei, M. & Klette, R. (2017). Computer vision for driver assistance. *Cham: Springer International Publishing*, **45**. 16

[64] Romano, D.M. (2003). *The nature of trust: conceptual and operational clarification*. Louisiana State University and Agricultural & Mechanical College. 3

[65] ROS, P. (2013). Robot operating system. *URL http://www. ros. org*. 22

[66] Salahat, E. & Qasaimeh, M. (2017). Recent advances in features extraction and description algorithms: A comprehensive survey. In *2017 IEEE international conference on industrial technology (ICIT)*, 1059–1063, IEEE. 15

[67] Sanders, T., Oleson, K.E., Billings, D.R., Chen, J.Y. & Hancock, P.A. (2011). A model of human-robot trust: Theoretical model development. In *Proceedings of the human factors and ergonomics society annual meeting*, vol. 55, 1432–1436, SAGE Publications Sage CA: Los Angeles, CA. 7

[68] Schneider, E., Sklar, E.I., Parsons, S. & Özgelen, A.T. (2015). Auction-based task allocation for multi-robot teams in dynamic environments. In *Towards Autonomous Robotic Systems: 16th Annual Conference, TAROS 2015, Liverpool, UK, September 8-10, 2015, Proceedings 16*, 246–257, Springer. 8

[69] Sheng, W., Jie, H., Yubin, L., Xuanchun, Y. & Yuhua, L. (2018). Influence of wing tip vortex on drift of single rotor plant protection unmanned aerial vehicle. *Nongye Jixie Xuebao/Transactions of the Chinese Society of Agricultural Machinery*, **49**. 9, 11

[70] Suzanne Barber, K., Goel, A. & Martin, C.E. (2000). Dynamic adaptive autonomy in multi-agent systems. *Journal of Experimental & Theoretical Artificial Intelligence*, **12**, 129–147. 5

[71] Szeliski, R. (2022). *Computer vision: algorithms and applications*. Springer Nature. 15

[72] Wei, W., Tan, L., Jin, G., Lu, L. & Sun, C. (2018). A survey of uav visual navigation based on monocular slam. In *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 1849–1853. 13

[73] Yasin, J.N., Mohamed, S.A., Haghbayan, M.H., Heikkonen, J., Tenhunen, H. & Plosila, J. (2020). Navigation of autonomous swarm of drones using translational coordinates. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, 353–362, Springer. 11